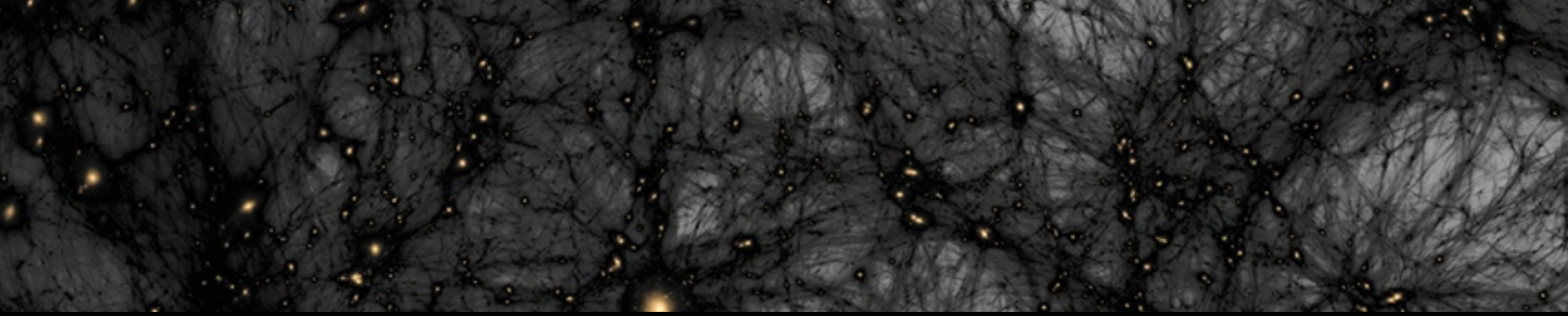


LECTURE IV: 21CM FORECASTS TUTORIAL WITH CAMB AND PYTHON

Alkistis Pourtsidou

Queen Mary University of London

https://github.com/Alkistis/Tonale2018

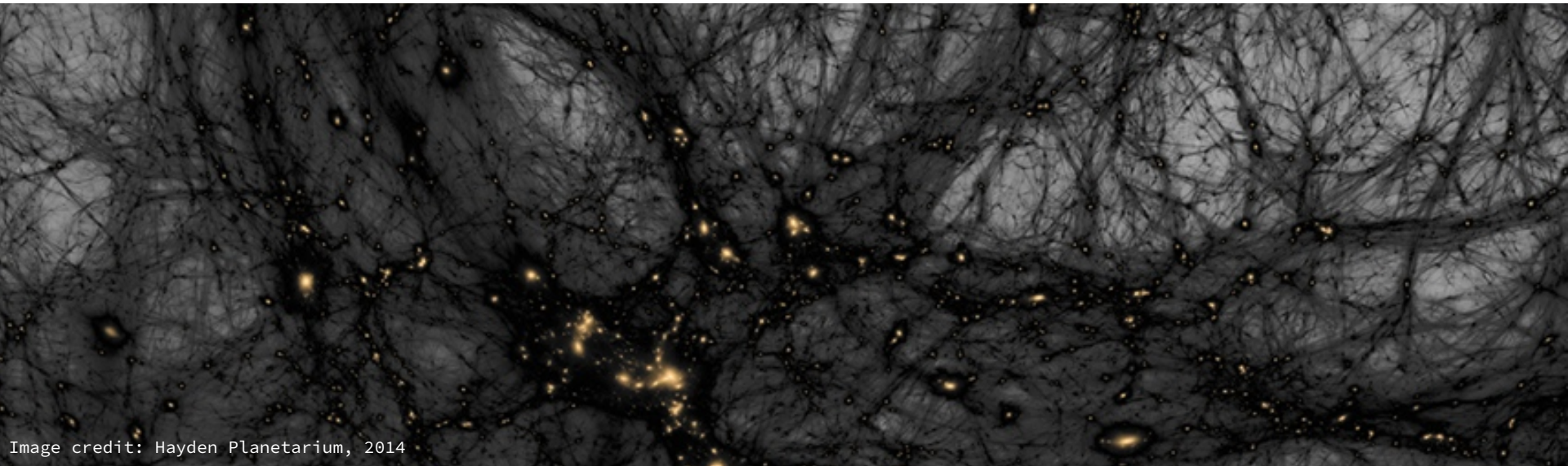
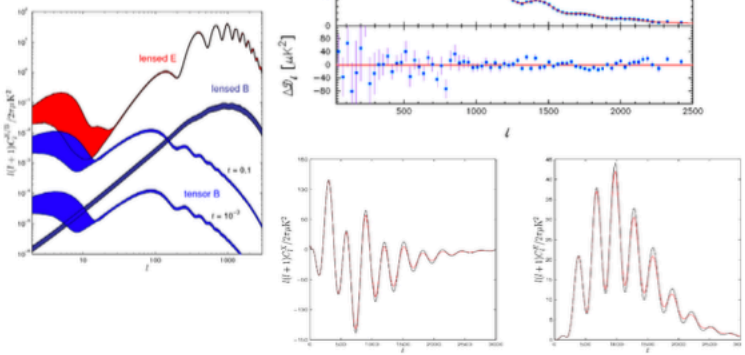


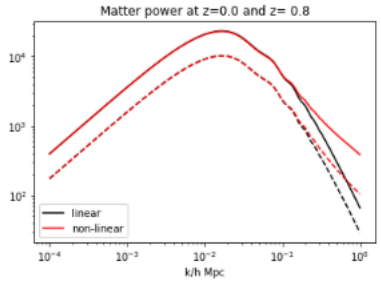
Image credit: Hayden Planetarium, 2014

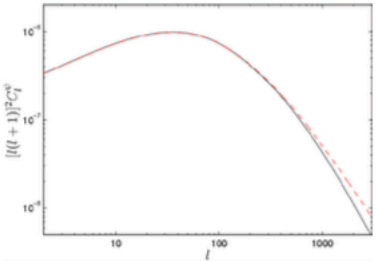
Slide from Antony Lewis

Introduction to CAMB

Code for Anisotropies in the Microwave Background







UNIVERSITY OF SUSSEX

Antony Lewis

http://cosmologist.info/

European Research Council

Established by the European Commission

Slide from Antony Lewis

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

THIS TUTORIAL

- In this tutorial we will use Jupyter notebooks and CAMB's python wrapper (installation instructions link in the notebooks)
- *The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualisations, and narrative text* **https://jupyter.org/index.html**
- **I strongly recommend installing Python and Jupyter using the Anaconda distribution http://jupyter.org/install**
- We start with **www.github.com/Alkistis/Tonale2018/CAMB-matter-power.ipynb**

Import numpy, matplotlib, and camb

```
import sys, platform, os

from matplotlib import pyplot as plt
import numpy as np

import camb
from camb import model, initialpower
```

- **matplotlib** is a Python 2D plotting library
- **NumPy** is the fundamental package for scientific computing with Python

Import scipy and interpolation

```
import scipy
from scipy.interpolate import interp1d
from __future__ import division
```

```
pi=np.pi
```

Set plotting and printing

```
%matplotlib inline

font = {'size'   : 16, 'family':'STIXGeneral'}
axislabelfontsize='x-large'
plt.rc('font', **font)
plt.rcParams['text.usetex'] = True
```

Cosmological parameters

In [5]:
```
c=3e5
hubble=0.678
omegab=0.022*pow(hubble,-2)
omegac=0.119*pow(hubble,-2)
om0=omegac+omegab
H00=100*hubble
Ass=2.14e-9
nss = 0.968

gamma=0.545
```

Set up the fiducial cosmology CAMB will use

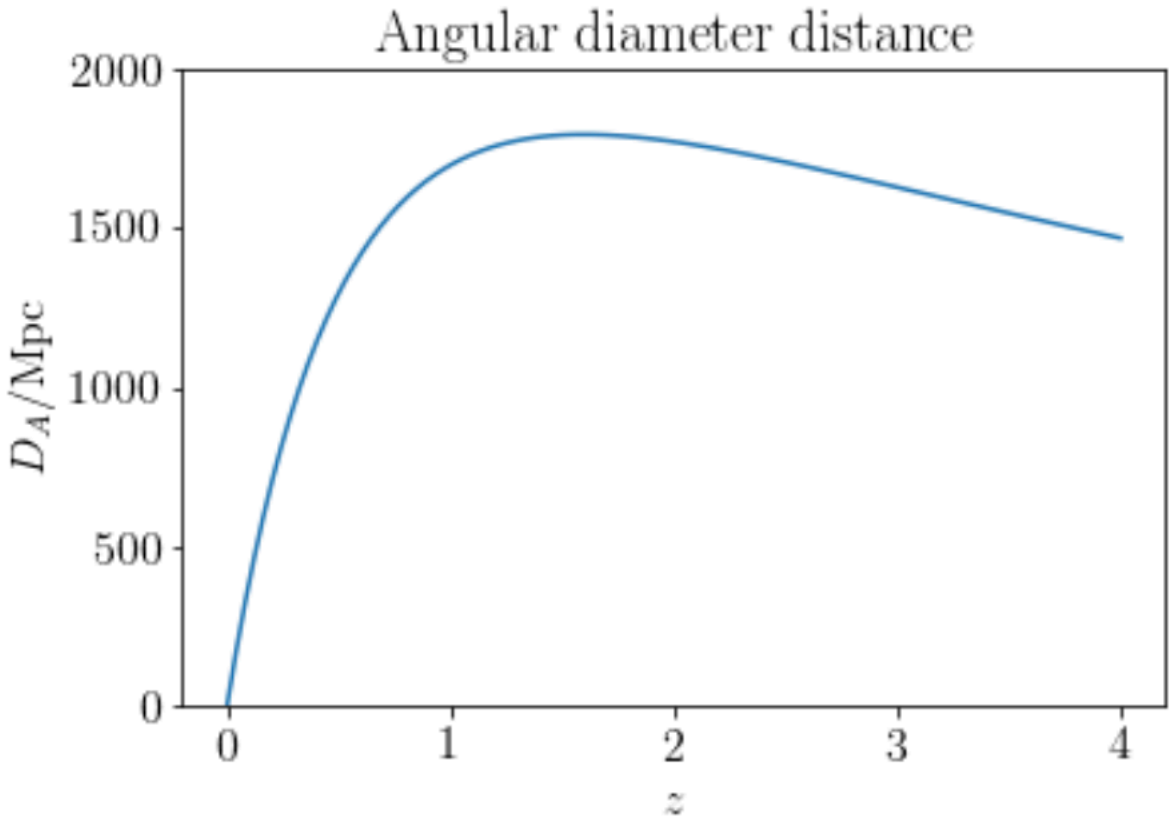
In [14]:
```
#Set up the fiducial cosmology
pars = camb.CAMBparams()
#Set cosmology
pars.set_cosmology(H0=H00, ombh2=omegab*pow(hubble,2),
                   omch2=omegac*pow(hubble,2),omk=0,mnu=0)
pars.set_dark_energy() #LCDM (default)
pars.InitPower.set_params(ns=nss, r=0, As=Ass)
pars.set_for_lmax(2500, lens_potential_accuracy=0);
```

Calculate results for these parameters

:
```
results = camb.get_results(pars)
#print pars
```

Get background quantities, e.g. the angular diameter distance

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



Get matter power spectrum using linear model or Halofit

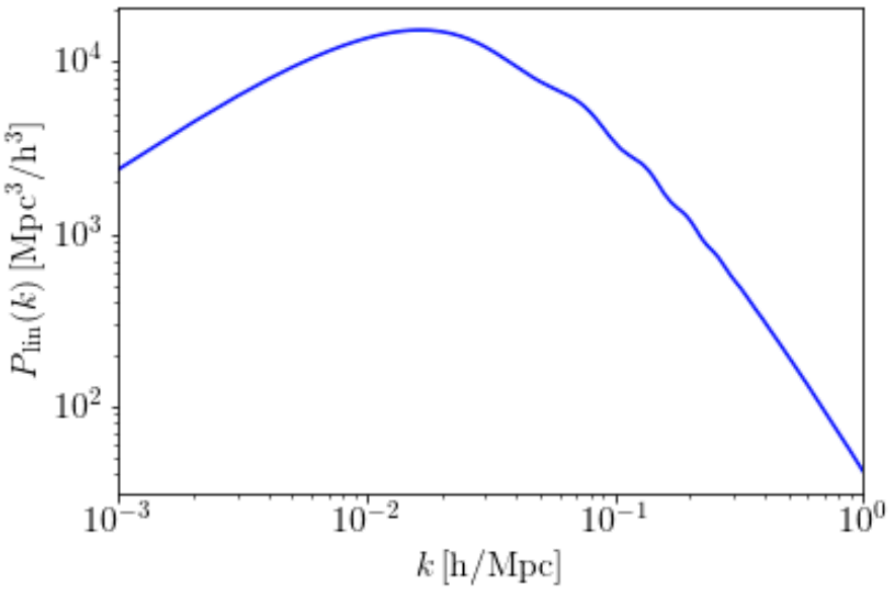
In [17]:
```
#Get matter power spectrum at some redshift
pars.set_matter_power(redshifts=[0.5], kmax=2.0)

#Linear spectra
pars.NonLinear = model.NonLinear_none
results.calc_power_spectra(pars)
kh, z, pk = \
         results.get_matter_power_spectrum(minkh=1e-4, maxkh=1, npoints = 200)

#Non-Linear spectra (Halofit)
pars.NonLinear = model.NonLinear_both
results.calc_power_spectra(pars)
kh_nonlin, z_nonlin, pk_nonlin = \
        results.get_matter_power_spectrum(minkh=1e-4, maxkh=1, npoints = 200)
```

Construct interpolating function for the linear case

```
Pk = interp1d(kh, pk)
```

Plot

```
kminn=1e-3
kmaxx=1.0

k1 = np.linspace(kminn,kmaxx,1000)

plin = [Pk(k) for k in k1]

plt.loglog(k1,plin,color='blue')

plt.xlim(kminn,kmaxx)

plt.xlabel("$k \, \mathrm{[h/Mpc]}$")
plt.ylabel("$P_{\, \mathrm{lin}}(k) \, \mathrm{[Mpc^3/h^3]}$")

plt.show()
```



The fisher matrix approach for forecasting

- Why do forecasts? And why with Fisher?
- Fisher forecasts very quick (compared to MCMC)
- Useful for survey strategy optimisations
- Or for figuring out if an observable is realistically going to be detected in your lifetime without spending too much resource…
- Fisher Matrix will give you the most optimistic constraints - it assumes you have access to all the statistical information (optimal weighting etc.)
- It doesn't usually contain any systematic effects, survey window functions etc. (but could try to include some of these in principle)

Slide from Licia Verde

Fisher matrix approach

(Fisher 1935) How well can a future experiment do?
(quick and easy but not always accurate)

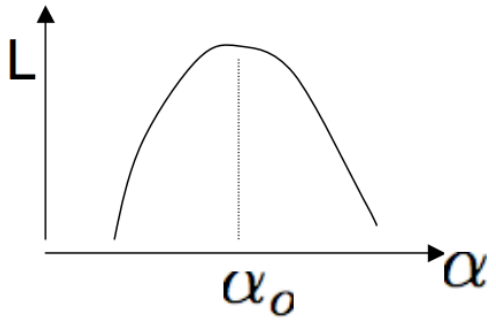
Fisher information matrix $F_{ij} = \left\langle \frac{\partial^2 L}{\partial\alpha_i \partial\alpha_j} \right\rangle$ $L = -\ln \mathcal{L}$

e.g. $$1/2(data - theory(\vec{\alpha}))C^{-1}(data - theory(\vec{\alpha})) \simeq$$
$$1/2(fiducial - theory(\vec{\alpha}))C^{-1}(fiducial - theory(\vec{\alpha}))$$

To develop intuition, one parameter case, Gaussian likelihood.

$L = 1/2(\alpha - \alpha_0)/\sigma_\alpha^2$

Second deriv. w.r.t. $\alpha \longrightarrow 1/\sigma_\alpha^2$



Slide from Licia Verde

Multi dimensional case…

$$\sigma^2_{\alpha_i,\alpha_j} \geq (\mathbf{F}^{-1})_{ij}$$ Parameters covariance

$$\sigma_{\alpha_i} \geq \sqrt{\frac{1}{F_{ii}}}$$ If all other parameters are fixed

$$\sigma_\alpha = (\mathbf{F}^{-1})^{1/2}_{ii}$$ Marginalized errors

Matrix inversion performed

21CM FISHER FORECAST EXAMPLE

- When we have the first auto-correlation HI intensity mapping data, we will try to measure the HI abundance - HI bias combination
- In the first round of data analysis, we will probably consider the cosmology fixed (e.g. LCDM Planck best-fit) and only measure the HI parameters
- We will forecast the expected constraints using www.github.com/Alkistis/Tonale2018/Fisher-omHIbHI.ipynb

$$P^{\rm HI} = \bar{T}_{\rm b}^2 b_{\rm HI}^2 P_m(k,z)$$

$$\bar{T}_{\rm b}(z) \propto \Omega_{\rm HI}(z)$$

FISHER MATRIX CALCULATION

$$F_{\rm ij} \approx \frac{1}{4\pi^2} \int_{k_{\rm min}}^{k_{\rm max}} k^2 dk \, [\partial_i \ln P^{\rm S} \partial_j \ln P^{\rm S}] V_{\rm eff}$$

$$V_{\rm eff} = V_{\rm sur} \left(\frac{P^{\rm S}}{P^{\rm S} + P^{\rm N}} \right)^2$$

- For more details see **https://arxiv.org/pdf/1610.04189.pdf**

We are going to work with the linear power spectrum (we won't use non-linear scales)

```
#Get matter power spectrum at z=0: P(k,z=0)
pars.set_matter_power(redshifts=[0.], kmax=2.0)

#Linear spectra
pars.NonLinear = model.NonLinear_none
results.calc_power_spectra(pars)
kh, z, pk = results.get_matter_power_spectrum(minkh=1e-4, maxkh=2.0, npoints = 200)
```

```
#Construct P(k,z=0) interpolating function, in units of Mpc (no h)
Pkz0 = interp1d(kh*hubble, pk[0]/pow(hubble,3))
```

IM experiment with redshift range 0.1 < z < 1.45

```
#Redshift bins
zlist = np.arange(0.1,1.45,0.1)
ztest = zlist[4]
Nzbins = len(zlist)

print (zlist)
print ("ztest =", ztest)
print ("Number of redshift bins =", Nzbins)
```

```
[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4]
ztest = 0.5
Number of redshift bins = 14
```

Useful functions to get the power spectrum in different z

```
#Define E(z) = H(z)/H0
def Ez(zc):
    return np.sqrt(1-om0+om0*pow(1+zc,3))

#Define the comoving distance
def drdz(zp):
    return (c/H00)/Ez(zp)
def rcom(zc):
    return scipy.integrate.romberg(drdz,0,zc)

print (rcom(ztest))
```

```
1946.7257776169436
```

```
#Define the growth function in LCDM
def fg(zz):
    omz=om0*pow(1+zz,3)/(om0*pow(1+zz,3)+1-om0)
    return pow(omz,gamma)

print (fg(ztest))
```

```
0.756249102937079
```

```
#Get the growth factor
def Dg_dz(zz):
    return fg(zz)/(1+zz)
def Dgz(zc):
    ans = scipy.integrate.romberg(Dg_dz, 0.0, zc)
    return np.exp(-ans)

print (Dgz(ztest))
```

```
0.7701490452523118
```

Fiducial HI abundance and bias fitting functions from SKA Cosmology Red Book 2018

```
def OmHI(zc):
    return 0.00048+0.00039*zc-0.000065*pow(zc,2)

def bHI(zc):
    return 0.67+0.18*zc+0.05*pow(zc,2)
```

Construct matter power spectrum P(k,z) - no RSDs

```
def Pkz(kk,zc):
    return pow(Dgz(zc),2)*Pkz0(kk)
```

Mean brightness temperature fitting function from SKA Cosmology Red Book 2018

```
def Tb(zc): #in mK
    return 0.0559+0.2324*zc-0.024*pow(zc,2)
```

Construct HI power spectrum - no RSDs

```
#Construct P_HI(k,z) [mK^2]
def PHI(kk,zc):
    return pow(Tb(zc),2)*pow(bHI(zc),2)*Pkz(kk,zc)
```

Assume a 4000 square degrees / 4000 hrs IM survey with MeerKAT and construct the noise power spectrum

```
Ndishes=64
Ddish=13.5*100 #cm
Nbeams=1

def thetab(zc):
    return 21*(1+zc)/Ddish

def omegapix(zc):
    return 1.13*pow(thetab(zc),2)

Area=4000.0 #deg^2
omegatot = Area*pow(pi/180,2)
ttotal = 4000*60*60*(Area/4000) #4000 hrs for 4000 deg^2

def fc(zc):
    return 1420.4/(1+zc)

def Tsky(zc):
    return 60*pow(300/fc(zc),2.55)*1e3

#receiver temperature from specs document
Tsyslist = [23.5*1e3,23.0*1e3,23.0*1e3,28.0*1e3,29.0*1e3,30.0*1e3,28.5*1e3,29.5*1e3,\
            31.0*1e3,33.0*1e3,34.0*1e3,35.0*1e3,37.0*1e3,38.0*1e3] #mK
```

```
def tobs(zc):
    return ttotal*(omegapix(zc)/omegatot)*Ndishes*Nbeams

Dzbin = 0.1
dfpix = 50*1e3 #Hz
midfreq = 1420.4e6 #Hz

def dzpix(zc):
    return pow(1+zc,2)*dfpix/midfreq
def sigpix(zc,Tsys):
    return Tsys/np.sqrt(dfpix*tobs(zc))
def dVpixdz(zz):
    return c*pow(rcom(zz),2)/(H00*Ez(zz))
def Vpix(zc):
    return omegapix(zc)*scipy.integrate.romberg(dVpixdz,zc-dzpix(zc)/2,zc+dzpix(zc)/2)

def Wsq(kk,zc): #very rough modelling, it should be Wsq(mu,kk,zc) - exercise!
    return np.exp(-pow(kk,2)*pow(rcom(zc),2)*pow(thetab(zc),2)/(8*np.log(2)))

def Pnoise(kk,zc,Tsys):
    return pow(sigpix(zc,Tsys),2)*Vpix(zc)*pow(Wsq(kk,zc),-1.)
```

Define minimum and maximum k

```
def kmin(zc):
    return 2*pi/np.sqrt(pow(rcom(zc),2)*omegatot)
def kmax(zc):
    return 0.14*pow(1+zc,2/(2+nss)); #non-linear cutoff Smith et al 2003

print (round(kmin(ztest),3), round(kmax(ztest),2))
```

```
0.003 0.18
```

Calculate effective volume for Fisher Matrix

```
#survey (bin) volume [Mpc^3]
def dVsurdz(zz):
    return omegatot*c*pow(rcom(zz),2)/(H00*Ez(zz))

def Vsur(zc):
    return scipy.integrate.romberg(dVsurdz,zc-Dzbin/2,zc+Dzbin/2)

#effective volume going in the Fisher matrix
def Veff(kk,zc):
    return Vsur(zc)*(PHI(kk,zc)/(PHI(kk,zc)+Pnoise(kk,zc,Tsys)))**2

print ("%.4g" % Vsur(ztest))
```

```
1.554e+09
```

- For parameter p, considering lnp directly gives fractional errors:

$$\sigma(\ln p) = \frac{\sigma(p)}{p}$$

Fisher matrix derivatives

```
def dlnP_dlnOmHIbHI(kk,zc):
    return 2.0
```

Fisher matrix

```
def dFdk(kk):
    return (1./(4*pi*pi))*pow(kk,2)*pow(dlnP_dlnOmHIbHI(kk,zc),2)*Veff(kk,zc)
```

Get the constraints on $\Omega_{\rm HI} b_{\rm HI}$

```
for i in range(0,Nzbins):
    zc = round(zlist[i],2)
    Tsys = Tsyslist[i]+Tsky(zc)
    K = np.linspace(kmin(zc), kmax(zc), 100)
    dF = dFdk(K)
    Fisher = scipy.integrate.simps(dF,K)
    #since we only vary one parameter, we don't need the covariance matrix
    print (zc, round(np.sqrt(1/Fisher),3))
```

```
0.1 0.01
0.2 0.005
0.3 0.005
0.4 0.007
0.5 0.009
0.6 0.011
0.7 0.013
0.8 0.015
0.9 0.018
1.0 0.022
1.1 0.026
1.2 0.03
1.3 0.036
1.4 0.041
```

INCLUDING REDSHIFT SPACE DISTORTIONS

- Neglecting RSDs, we have seen that: $\bar{T}_{\rm b}(z) \propto \Omega_{\rm HI}(z)$

$$P^{\rm HI} = \bar{T}_{\rm b}^2 b_{\rm HI}^2 P_m(k,z)$$

- Including RSDs, we can break the degeneracy between HI abundance and bias **[Masui et al 2013, AP et al 2017]**

The full HI signal power spectrum in redshift space can be written as

$$P^{\rm S} \equiv P^{\rm HI}(k,z;\mu) = \bar{T}_b^2 b_{\rm HI}^2 [1+\beta_{\rm HI}(z)\mu^2]^2 P(k,z)$$

where P is the matter power spectrum, $b_{\rm HI}$ the HI bias, $\mu = \hat{k}\cdot\hat{z}$ and $\beta_{\rm HI}$ the redshift space distortion parameter equal to $f/b_{\rm HI}$ in linear theory, where $f \equiv d{\rm ln}D/d{\rm ln}a$ is the linear growth rate with the scale factor $a = 1/(1+z)$.

BREAKING THE DEGENERACY

- Including RSDs, we can break the degeneracy between HI abundance and bias [Masui et al 2013, AP et al 2017]
- **You can try it as an exercise!**