powerspectrum_utilstest

Unknown Author

April 7, 2014

0.1 Purpose: Conveniently obtaining the linear power spectrum (currently only from) CAMB outputs:

- Method: "CAMBoutfile"
  - from power spectrum output at desired redshift
  - from transfer function output at desired redshift and a cosmological model with Amplitude specified
  - from transfer function output at desired redshift and a cosmological model with sigma8 specified at the same time
- Method: "CAMBoutgrowth"
  - from transfer function output at z = 0 and a cosmological model with sigma8 at z =0, obtain power spectrum at z < 0.
  - from power spectrum output at z = 0, and a cosmological model, obtain the power spectrum at z < 0.
  - from the transfer function output at z =0, and a cosmological model , As, obtain power spectrum at z < 0.

0.2 Some Options:

- matter power spectrum: matter = CDM, Baryons, Massive neutrinos
- cb power spectrum = CDM , baryons only.
- Normalization: is normalized to sigma8 for the cb part or the total matter fluctuations
- Obtain the logarithmically or inearly interpolated power spectrum at a list of wave numbers. For wavenumbers supplied outside the range of data points, the extrpolated values are given by np.nan, by default.

In [144]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import camb_utils.cambio as cio
import psutils as psu
from interfaces import FCPL
import utils.plotutils as pu
import matplotlib.gridspec as gridspec
import matplotlib.ticker as ticker
#plt.ticklabel_format(style='sci',axis='x',scilimits=(0,0))
#majorformatter = ticker.ScalarFormatter(useOffset =False)
```

In [2]:
```
# INPUT FILES (CAMB outputs)
dirn  = "./"
#M000 (LCDM)
tkfile  = dirn +"example_data/M000/m000n0_transfer_fin_out.dat"
pkfile = dirn + "example_data/M000/m000n0_matterpower_fin.dat"
#M000n1 (same LCDM, with some fraction of CDM replaced by massive neutrinos)
```

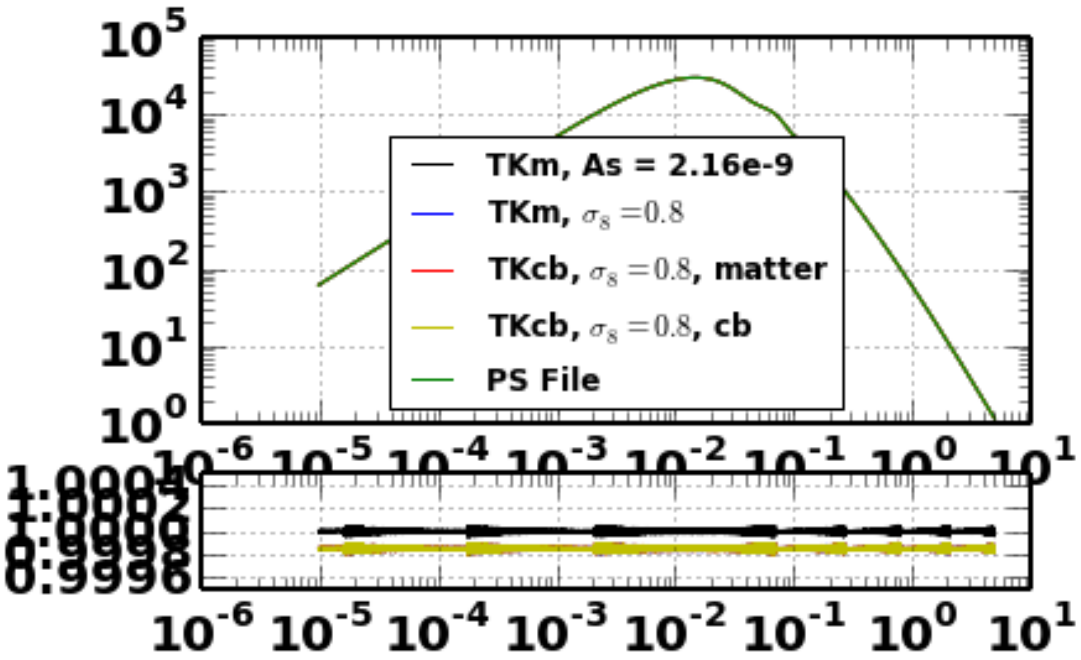
```
ntkfile = dirn +"example_data/M000n1/m000n1_transfer_fin_out.dat"
npkfile = dirn +"example_data/M000n1/m000n1_matterpower_fin.dat"
```

In [3]:
```
#Cosmological Models:
#               Same cosmology represented differently
M000 = FCPL(H0 = 71. ,Om0  = 0.26479, Ob0 = 0.044793, ns = 0.963, As = 2.16e-9)
M000s  = FCPL(H0 = 71., Om0 = 0.26479, Ob0 = 0.044793, ns = 0.963, sigma8 = 0.8)
#
M000n1 = FCPL(H0 = 71., Om0 = 0.26479, Ob0 = 0.044793, ns = 0.963, sigma8 = 0.8, sigma
```

1 Results for M000 (LCDM)

In [4]:
```
        #powerspectrum
#use power spectrum from CAMB power spectrum output
psfrompk = psu.powerspectrum(koverh = None, asciifile =pkfile)
#use matter power spectrum from CAMB transfer function output, with As
psfromtkm = psu.powerspectrum(koverh = None, pstype ="matter", asciifile =tkfile, cosm
#use matter power spectrum from CAMB transfer function output, with sigma8
psfromtkms = psu.powerspectrum(koverh = None, pstype ="matter", sigma8type = "matter",
#use cb power spectrum from CAMB transfer function output, normalized to matter
psfromtkcbs = psu.powerspectrum(koverh = None, pstype ="cb", sigma8type = "matter", as
#use cb power spectrum from CAMB transfer function output, normalized to cb
psfromtkcbscb = psu.powerspectrum(koverh = None, pstype ="cb", sigma8type = "cb", asci
#
#psfrompkn = psu.powerspectrum(koverh = None, asciifile  = npkfile)
#psfromtknm = psu.powerspectrum(koverh = None, asciifile  = ntkfile, pstype ="matter",
```

In [5]:
```
#Want to check results to 1 percent
majorformatter = ticker.ScalarFormatter(useOffset =False)
pscompfig, pscomp_ax0, pscomp_ax1 = pu.settwopanel(setdifflimits = [0.9995,1.0005])
pscomp_ax0.loglog(psfromtkm[0], psfromtkm[1], label = "TKm, As = 2.16e-9")
pscomp_ax0.loglog(psfromtkms[0], psfromtkms[1], label = r'TKm, $\sigma_8 =0.8$')
pscomp_ax0.loglog(psfromtkcbs[0], psfromtkcbs[1], label = "TKcb, $\sigma_8 =0.8$, matt
pscomp_ax0.loglog(psfromtkcbs[0], psfromtkcbscb[1], label = "TKcb, $\sigma_8 =0.8$, cb
pscomp_ax0.loglog(psfrompk[0],psfrompk[1],label="PS File")
pscomp_ax0.legend(loc="lower center")
pscomp_ax1.plot(psfromtkm[0],psfromtkm[1]/psfrompk[1])
pscomp_ax1.plot(psfromtkm[0],psfromtkms[1]/psfrompk[1])
pscomp_ax1.plot(psfromtkcbs[0], psfromtkcbs[1]/psfrompk[1], label = "TKcb, $\sigma_8 =
pscomp_ax1.plot(psfromtkcbs[0], psfromtkcbscb[1]/psfrompk[1], label = "TKcb, $\sigma_8
pscomp_ax1.yaxis.set_major_formatter(majorformatter)
pscomp_ax1.set_xscale('log')
#ppscomp_ax1.get_yticklabels()
#pscomp_ax1.set_yscale('log')
```

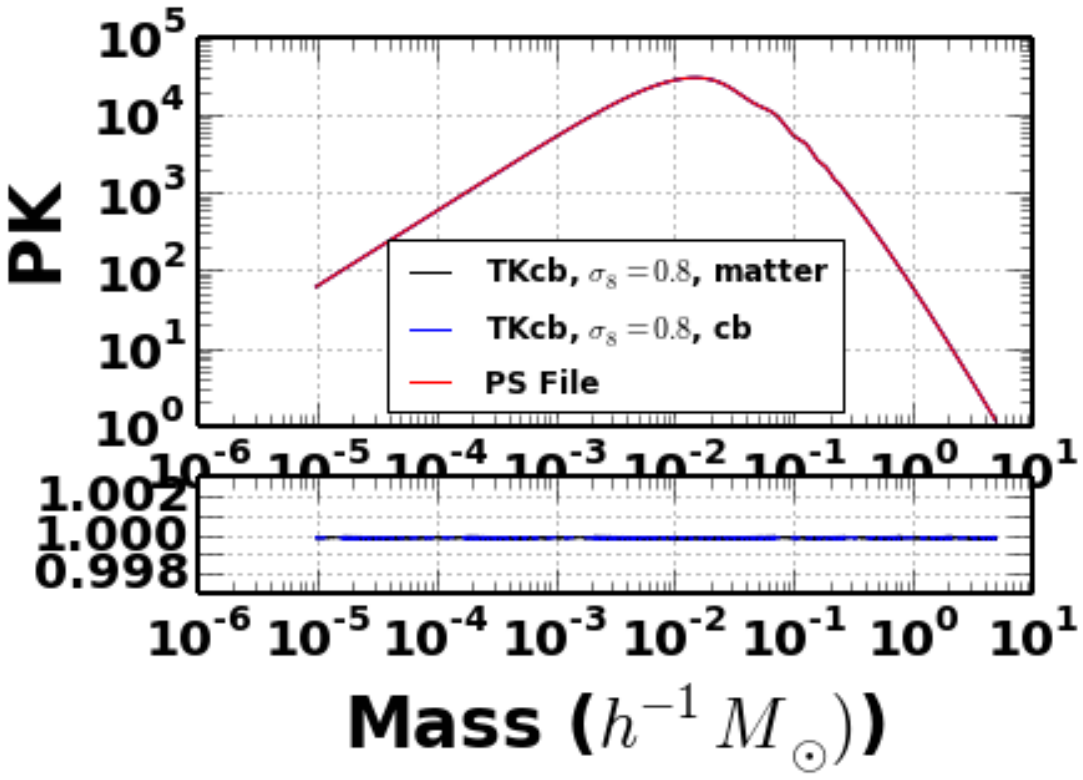


Caption: Comparison of the different modes of computing power spectrum for LCDM models. These should be identical, and are very nearly so, as shown by the next plot in greater detail.Check only the cb components:

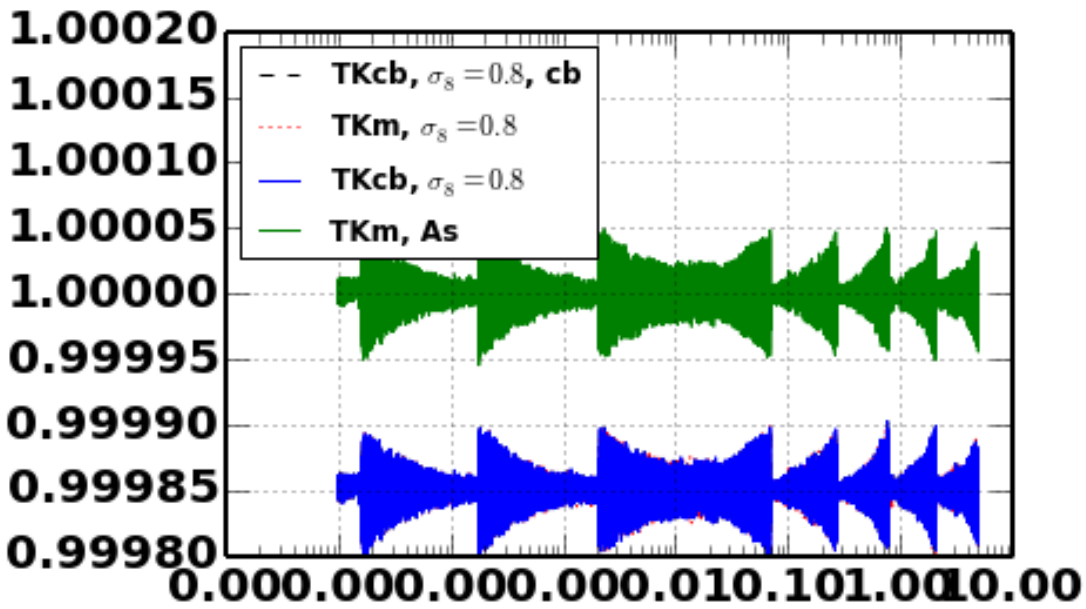
In [8]:
```
cbfig, cb_ax0, cb_ax1 = pu.settwopanel(setdifflimits = [0.997,1.003])
cb_ax0.loglog(psfromtkcbs[0], psfromtkcbs[1], label = "TKcb, $\sigma_8 =0.8$, matter")
cb_ax0.loglog(psfromtkcbs[0], psfromtkcbscb[1], label = "TKcb, $\sigma_8 =0.8$, cb")
cb_ax0.loglog(psfrompk[0],psfrompk[1],label="PS File")
cb_ax0.legend(loc="lower center")
cb_ax0.set_ylabel("PK ")
cb_ax1.plot(psfromtkcbs[0], psfromtkcbs[1]/psfrompk[1], label = "TKcb, $\sigma_8 =0.8$
cb_ax1.plot(psfromtkcbs[0], psfromtkcbscb[1]/psfrompk[1],ls='dashed', label = "TKcb, $
cb_ax1.set_xscale('log')
plt.setp(cb_ax1.get_yticklabels()[1::2],visible=False)
cb_ax1.set_xlabel(r'Mass ($h^{-1} M_\odot)$)')
#cb_ax1.set_ylabel("PK / PK (from pk)")
```

Out [8]:

```
<matplotlib.text.Text at 0x108353950>
```



In [9]:
```
#Only the case where As is provided is at 1.0000, all other cases where sigma8 is calc
#are lower
plt.plot(psfromtkcbs[0], psfromtkcbscb[1]/psfrompk[1],'k',ls='dashed', label = "TKcb,
plt.plot(psfromtkcbs[0], psfromtkms[1]/psfrompk[1],'r',ls='dotted', label = "TKm, $\si
plt.plot(psfromtkcbs[0], psfromtkcbs[1]/psfrompk[1],'b-' ,label = "TKcb, $\sigma_8 =0.
plt.plot(psfromtkcbs[0], psfromtkm[1]/psfrompk[1],'g-' ,label = "TKm, As")
plt.xscale('log')
plt.legend(loc="best")
plt.grid(True)
plt.ylim(0.9998,1.0002)
import matplotlib.ticker as ticker
#plt.ticklabel_format(style='sci',axis='x',scilimits=(0,0))
majorformatter = ticker.ScalarFormatter(useOffset =False)
ax = plt.gca()
ax.yaxis.set_major_formatter(majorformatter)

ax.xaxis.set_major_formatter(majorformatter)
#plt.ticklabel_format(useOffset =False)
```



Caption: For LCDM models, cb and matter power spectrum are the same thing. Reproducing the CAMB power spectra is limited only by our σ_8 integrals, which matches CAMB power spectra to an order similar to numerical noise. Whenever the amplitude of the power spectrum is supplied (ie. we do not have to calculate σ_8 to fix the amplitude, we get something that lies under the green curve. If we have to obtain σ_8, we get something that lies under the blue curve.

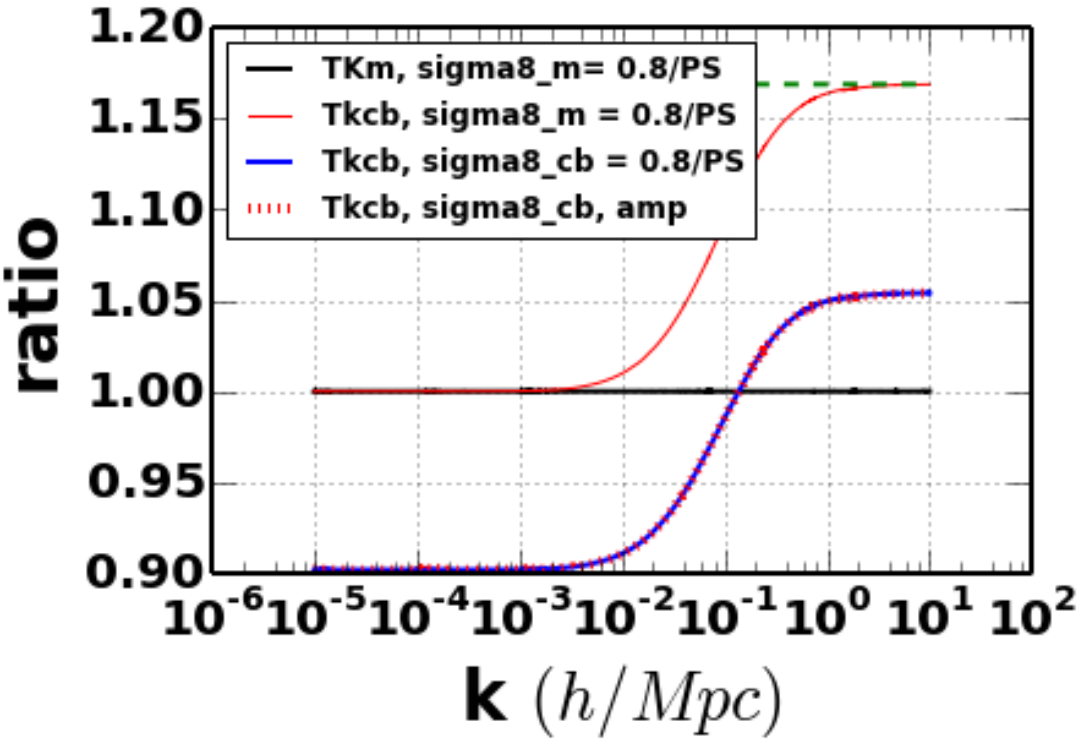
2 Model with neutrinos (M000n1)

In [218]:
```
#Model With Neutrinos
psfrompkn = psu.powerspectrum(koverh = None, asciifile  = npkfile)#power spectrum outp
psfromtknm = psu.powerspectrum(koverh = None, asciifile  = ntkfile, pstype ="matter",
psfromtkncbs = psu.powerspectrum(koverh = None, asciifile  = ntkfile, pstype ="cb",sig
psfromtkncbscb = psu.powerspectrum(koverh = None, asciifile  = ntkfile, pstype ="cb",s
```

In [15]:
```
plt.plot(psfrompkn[0],psfromtknm[1]/psfrompkn[1],'k-',lw=2.0, label = "TKm, sigma8_m=
plt.plot(psfrompkn[0],psfromtkncbs[1]/psfrompkn[1],'r-',label ="Tkcb, sigma8_m = 0.8/P
plt.plot(psfrompkn[0],psfromtkncbscb[1]/psfrompkn[1],'b',label ="Tkcb, sigma8_cb = 0.8
plt.plot(psfrompkn[0],3.704/4.107*psfromtkncbs[1]/psfrompkn[1],"r",ls= "dotted",lw =4.
#Cannot predict the change in As, but show that the red curve is expected if I know wh
M000n1_rhom = M000n1.Oc0 +M000n1.Ob0 + M000n1.On0
M000n1_rhocb = M000n1.Oc0 + M000n1.Ob0
f = M000n1_rhom**2/M000n1_rhocb**2
plt.plot(psfrompkn[0],np.ones(len(psfrompkn[0]))*f,'g',ls='dashed', lw = 2)
#Assume rho_nu totally unclustered, then only contribution comes from
#the denominator in delta rho/rho
#Shown in green dashed line
plt.legend(loc="upper left")
plt.xscale('log')
plt.grid(True)
plt.xlabel(r'k $(h/Mpc)$')
plt.ylabel(r'ratio')
```

Out [15]:

```
<matplotlib.text.Text at 0x10b775fd0>
```



Caption: Ratio of power spectra for M000n1 model at z = 0. The denominator is always the CAMB power spectrum output (matter power spectrum) at z = 0 of M000n1. Tkm (thick black curve) is the power spectrum calculated from the total transfer function and a sigma8 = 0.8, so this ratio should be 1. Tkcb, sigma8_m = 0.8 (thin red solid) is the cb power spectrum, normalized so that the total power spectrum has sigma8 = 0.8. The amplitude of fluctuations is the same as for the matter power spectrum, and the curve should be the same as the matter power spectrum at low k. At high k where the neutrinos are essentially unclustered, this cb power spectrum is expected to be ((rhocb + rhonu)/rhocb)^2 (green dashed line), and works asymptotically. The cb power spectrum (solid blue line), normalized so that the cb components have a σ_8 of 0.8 have a different amplitude of fluctuations from the matter power spectrum. Since Pcb ($\sigma_8(matter) = 0.8$) is above the matter power spectrum, this amplitude is smaller than the amplitude of fluctuations for M000n1, but should have the same shape as Pcb ($\sigma_8(matter) = 0.8$) Pcb. The (blue-dashed) curve is obtained from the Pcb (sigma_8(m) =0.8) curve by multiplying the curve by ratio of the amplitude of the power spectrum for the solid blue curve (calculated) to the amplitude for the total matter power spectrum (calculated). Therefore we understand this entire set of plots

2.1 Suppression of Power spectrum due to Neutrinos

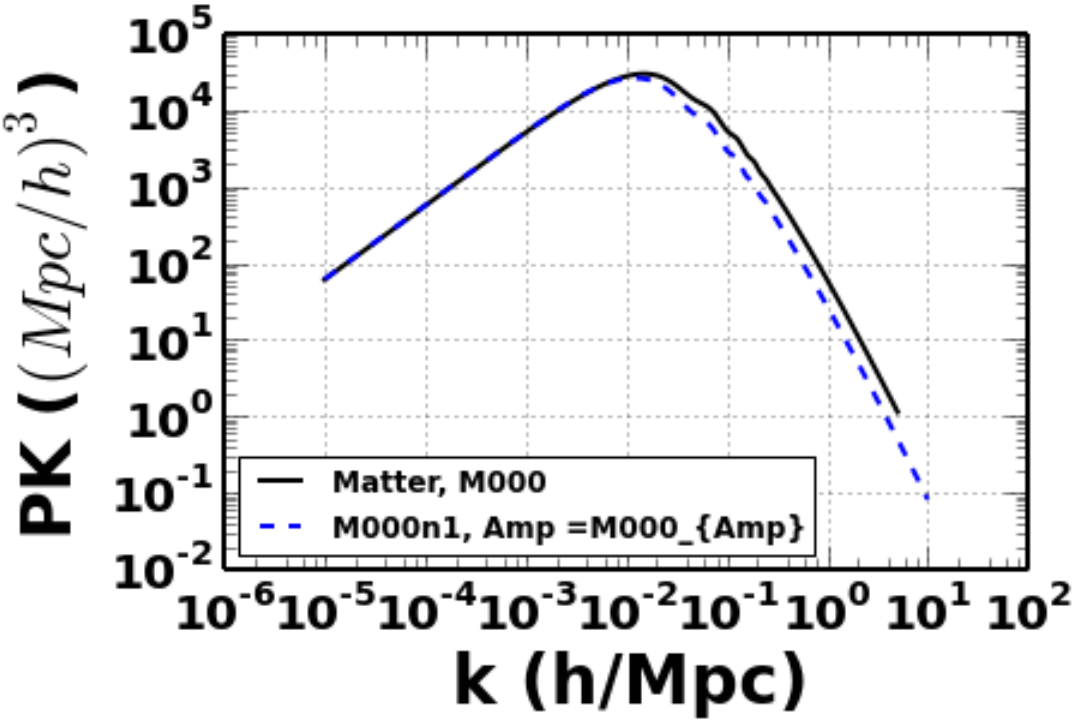
In [23]:

```
psfrompkint = psu.powerspectrum(koverh = psfromtknm[0], asciifile =tkfile,cosmo =M000s
plt.loglog(psfrompkint[0],psfrompkint[1],'k-',lw =2.0, label = "Matter, M000")
plt.loglog(psfromtknm[0],psfromtknm[1]*2.16/4.074,'b--',lw = 2.0, label = "M000n1, Amp
plt.grid(True)
plt.legend(loc="best")
plt.xlabel("k (h/Mpc)")
plt.ylabel("PK ($(Mpc/h)^{3}$)")
```

```
#print len(psfrompkint[0])
#print len(psfromtknm[0])
#print psfrompkint
```

Out [23]:

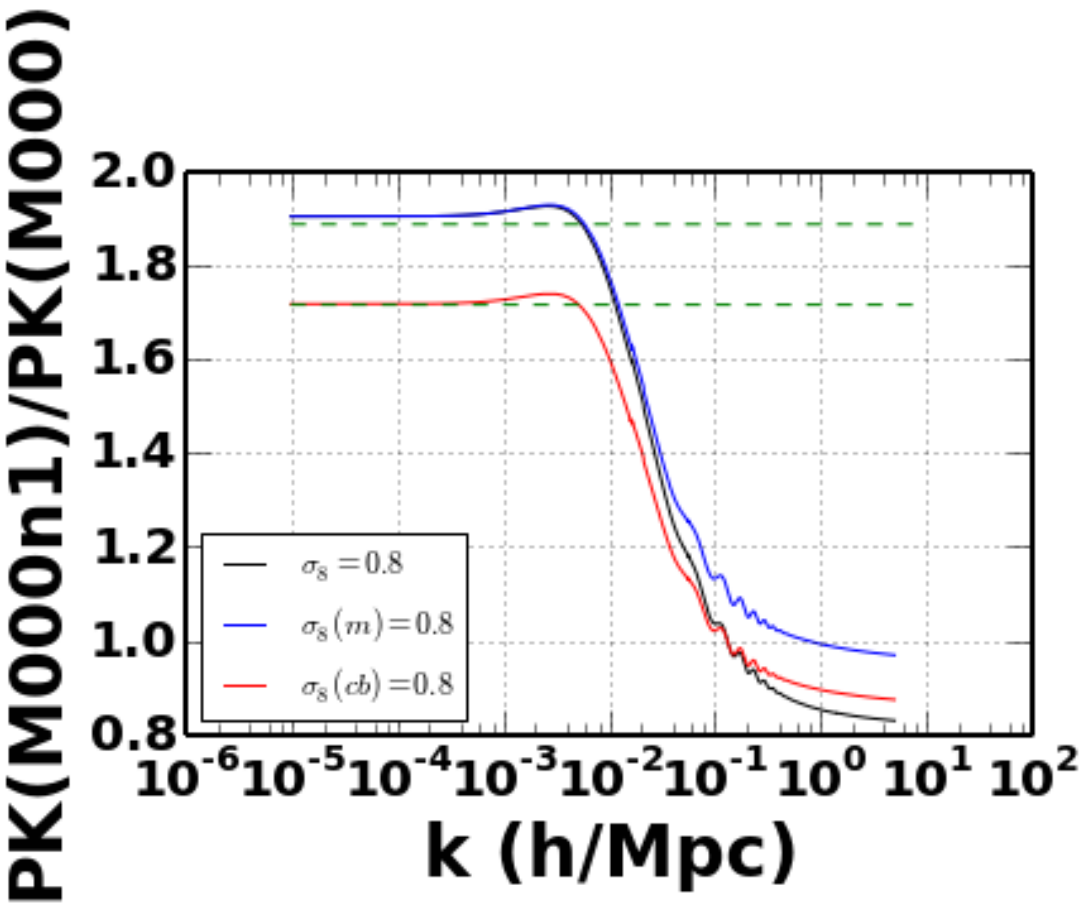
```
<matplotlib.text.Text at 0x10ca71d10>
```



Caption : Shows the matter power spectrum computed for M000 and M00n1 with the amplitude adjusted to be the same as the amplitude for M000.

In [37]:

```
ncomp_ax1 = plt.subplot()
ncomp_ax1.plot(psfromtknm[0],psfromtknm[1]/psfrompkint[1], label   ="$\sigma_8=0.8$")
ncomp_ax1.plot(psfromtknm[0],psfromtkncbs[1]/psfrompkint[1], label ="$\sigma_8 (m) = 0
ncomp_ax1.plot(psfromtknm[0],psfromtkncbscb[1]/psfrompkint[1], label = "$\sigma_8 (cb)
ncomp_ax1.plot(psfromtknm[0],np.ones(len(psfromtknm[0]))*3.704/2.16,"g", ls = 'dashed'
ncomp_ax1.plot(psfromtknm[0],np.ones(len(psfromtknm[0]))*4.074/2.16,"g", ls = 'dashed'
#ncomp_ax1.plot(psfromtknm[0],np.ones(len(psfromtknm[0]))*0.8*f,"r", ls = 'dashed')
ncomp_ax1.legend(loc="lower left")
ncomp_ax1.set_ylabel("PK(M000n1)/PK(M000)")
ncomp_ax1.set_xlabel("k (h/Mpc)")
plt.grid(True)
ncomp_ax1.set_xscale('log')
```



Caption: Ratio of the linear power spectrum of the model M000n1 normalized in three different ways to the M000 power spectrum showing the suppression of the high k power spectrum relative to the low k par. The normalizations are chosen so that (black, solid) usual matter power spectrum, (blue, solid) cb power spectrum normalized so that the matter power spectrum has $\sigma_8 = 0.8$, and (red, solid) cb power spectrum normalized so that the cb power spectrum has $\sigma_8 = 0.8$. The behavior of the three lines at low k is understood in terms of the amplitudes of fluctuations required for the normalizations. As seen in the ratio of different normalizations to the M000n1 matter power spectrum, the cb power spectrum normalized so that cb $\sigma_8 = 0.8$ has a smaller amplitude and should be below the the cb power spectrum normalized so that cb power spectrum has a $\sigma_8 = 0.8$. Thus the red curve is always below the blue curve with a constant ratio. The low k asymoptotics is given by the relative amplitudes and the expected values are shown by the green dashed curves. The relative behavior of the black, blue, and red curves can be better understood from the previous curve showing the ratios. However, the supression of any of these curves with respect to their low k values cannot be understood in terms of such numbers which are post-processing/counting issues, while the supression is a real effect due to lack of sourcing of the Poisson equation. In comparison, the effect of sourcing due to the clustering of neutrinos is negligible as seen in the power spectrum paper.

3 Power Spectra at different redshifts:

In [6]:
```
#M000files
#transfer and power spectrum files at redshift 1, 2, 3
tkfile1 = dirn +"example_data/M000/m000n0_transfer_247_out.dat"
pkfile1 = dirn +"example_data/M000/m000n0_matterpower_247.dat"
pkfile2 = dirn +"example_data/M000/m000n0_matterpower_163.dat"
```

```
pkfile3 = dirn + "example_data/M000/m000n0_matterpower_121.dat"

#M000n1 files
#power spectrum file at redshift  1, 2, 3
ntkfile1 = dirn +"example_data/M000n1/m000n1_transfer_247_out.dat"
npkfile1 = dirn +"example_data/M000n1/m000n1_matterpower_247.dat"
npkfile2 = dirn +"example_data/M000n1/m000n1_matterpower_163.dat"
npkfile3 = dirn + "example_data/M000n1/m000n1_matterpower_121.dat"


psfromtk1 = psu.powerspectrum(koverh = None, asciifile = tkfile1, cosmo = M000)
psfrompk1 = psu.powerspectrum(koverh = None, asciifile = pkfile1, cosmo = M000)
psfromtk10 = psu.powerspectrum(koverh = None, asciifile = tkfile, cosmo = M000s,z =1.0
psfromtk2  = psu.powerspectrum(koverh = None, asciifile = pkfile, cosmo = M000,z =2.0,
psfromtk2A = psu.powerspectrum(koverh = None, asciifile = tkfile, cosmo = M000,z =2.0,

M000psfrompk1  = psu.powerspectrum(koverh = None, asciifile = pkfile1)
M000psfrompk2  = psu.powerspectrum(koverh = None, asciifile = pkfile2)
M000psfrompk3  = psu.powerspectrum(koverh = None, asciifile = pkfile3)
M000psfromtk10 = psu.powerspectrum(koverh = None, asciifile = tkfile, cosmo = M000s, z
M000psfromtk20 = psu.powerspectrum(koverh = None, asciifile = tkfile, cosmo = M000s, z
M000psfromtk30 = psu.powerspectrum(koverh = None, asciifile = tkfile, cosmo = M000s, z

M000n1psfrompk1  = psu.powerspectrum(koverh = None, asciifile = npkfile1)
M000n1psfrompk2  = psu.powerspectrum(koverh = None, asciifile = npkfile2)
M000n1psfrompk3  = psu.powerspectrum(koverh = None, asciifile = npkfile3)
M000n1psfromtk10 = psu.powerspectrum(koverh = None, asciifile = ntkfile, cosmo = M000n
M000n1psfromtk20 = psu.powerspectrum(koverh = None, asciifile = ntkfile, cosmo = M000n
M000n1psfromtk30 = psu.powerspectrum(koverh = None, asciifile = ntkfile, cosmo = M000n

mykoverh = M000n1psfrompk1[0]
iM000psfrompk1  = psu.powerspectrum(koverh = mykoverh, asciifile = npkfile1)
iM000psfrompk2  = psu.powerspectrum(koverh = mykoverh, asciifile = npkfile2)
iM000psfrompk3  = psu.powerspectrum(koverh = mykoverh, asciifile = npkfile3)
iM000psfromtk10 = psu.powerspectrum(koverh = mykoverh, asciifile = tkfile, cosmo = M00
iM000psfromtk20 = psu.powerspectrum(koverh = mykoverh, asciifile = tkfile, cosmo = M00
iM000psfromtk30 = psu.powerspectrum(koverh = mykoverh, asciifile = tkfile, cosmo = M00
```

3.1 Growth Functions:

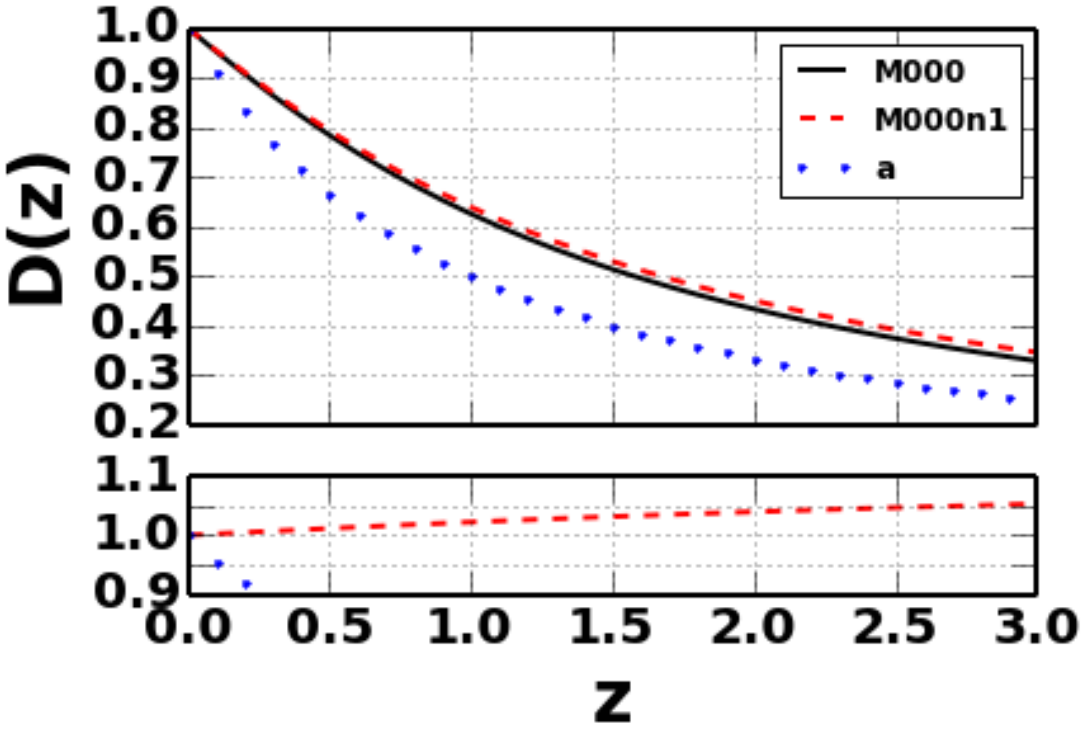
The growth function for the models are different. M000 has a scale independent growth function, while M000n1 has a scale-dependent growth function whose low k limit approximately (Note that the M000 model still has 3.04 massless neutrino species, while the massive neutrino model does not . . . it was run with 0.046 massless neutrinos, which would better have been 0. with a higher temperature) coincide with M000 (for small redshifts, where the massive neutrinos behave as CDM). In the high k limit, the neutrinos do not contribute to the clustering.

In [172]:
```
z = np.arange(0.00,3.1,0.1)
gfig, gax0, gax1 = pu.settwopanel(setdifflimits =[0.9,1.1])
gax0.plot ( z, M000.growth(z) [0], 'k-', lw =2.0, label = "M000")
gax0.plot (z, M000n1.growth(z)[0], 'r--',label = "M000n1", lw =2 )
gax0.plot(z , 1./(1+z), 'b.',label = "a")
gax0.set_ylabel("D(z)")
gax0xticks  = gax0.axes.get_xticklabels()
gax0.set_xticklabels(gax0xticks[::2], visible= False)
gax0.legend(loc="best")
gax1.plot(z, (1./M000.growth(z)[0]*M000n1.growth(z)[0]),'r--',lw =2.0)
gax1.plot(z, 1./M000.growth(z)[0]/(1.+z), 'b.')
gax1ticks = gax1.axes.get_yticklabels()
gax1.set_xlabel("z")
gax1.yaxis.set_ticks([0.9,0.95,1.0, 1.05, 1.1])
gax1.yaxis.set_ticklabels([0.9,"",1.0,"",1.1])
#gax1.set_yticklabels([0.95,1.0], visible= True)
#print len(gax1ticks)
```

```
#plt.setp(gax1ticks[::,2],visible=True)
#
```

Out [172]:

```
[<matplotlib.text.Text at 0x10ec67f10>,
 <matplotlib.text.Text at 0x10ec60f50>,
 <matplotlib.text.Text at 0x10ec82dd0>,
 <matplotlib.text.Text at 0x10ec80710>,
 <matplotlib.text.Text at 0x10ec58ed0>]
```

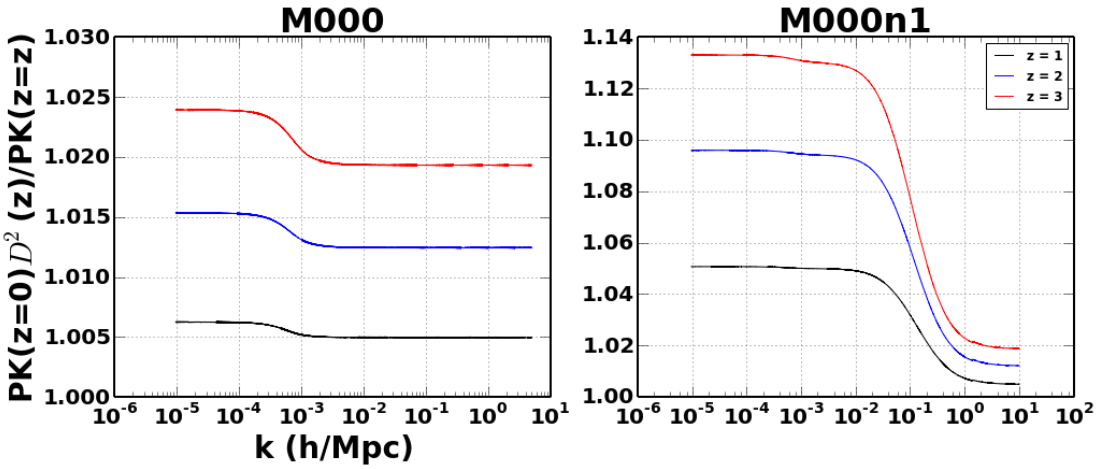


Caption : The growth function of the model with neutrinos is higher than the growth function in a similar model without neutrinos: in order to obtain the same amplitude of a fluctuation at z = 1.0, the fluctuation should have been larger at a higher z for the model with neutrinos, because with the absence of some dark matter the fluctuation will grow less. The ratio reaches about 5 percent at z = 3.0. We will see this effect on the high redshift power spectra obtained from the z = 0, power spectra scaled by the growth function. The blue dotted curve shows the growth function that would have been obtained in a massless neutrino EdS model with no dark energy, showing that dark energy slows the clustering of matter appreciably.

In [164]:

```
fig = plt.figure(figsize = (16,6))

ax = plt.subplot(1,2,1)
ay = plt.subplot(1,2,2)


ax.yaxis.set_major_formatter(majorformatter)
ax.plot(M000psfrompk1[0],M000psfromtk10[1]/M000psfrompk1[1])
ax.plot(M000psfrompk2[0],M000psfromtk20[1]/M000psfrompk2[1])
ax.plot(M000psfrompk2[0],M000psfromtk30[1]/M000psfrompk3[1])
#ax.plot(M000psfrompk1[0],M000psfromtk10[1]/M000psfrompk1[1])
#ax.plot(M000psfrompk2[0],M000psfromtk20[1]/M000psfrompk2[1])
#ax.plot(M000psfrompk2[0],M000psfromtk30[1]/M000psfrompk3[1])
```

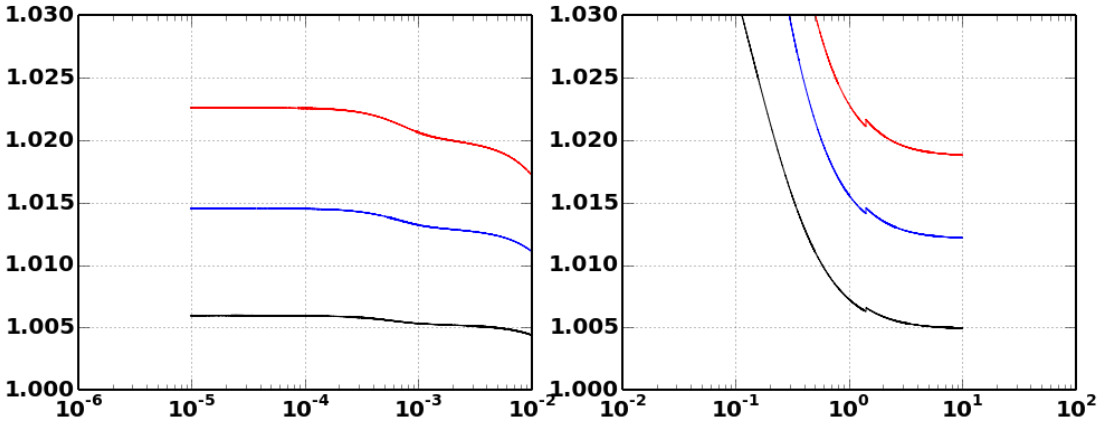
```
ax.grid(True)
ax.set_xlabel("k (h/Mpc)")
ax.set_ylabel("PK(z=0)$D^2$(z)/PK(z=z)")
ax.set_ylim(1.,1.03)
ax.set_title("M000")
ax.set_xscale('log')
#
ay.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/M000n1psfrompk1[1],label = "z = 1")
ay.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/M000n1psfrompk2[1],label = "z = 2")
ay.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/M000n1psfrompk3[1],label = "z = 3")
#ay.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/M000n1psfrompk1[1])
#ay.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/M000n1psfrompk2[1])
#ay.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/M000n1psfrompk3[1])
ay.set_xscale('log')
ay.legend(loc="best")
ay.set_title("M000n1")
ay.grid(True)

#az.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/M000n1psfrompk1[1])
#az.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/M000n1psfrompk2[1])
#az.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/M000n1psfrompk3[1])
#az.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/iM000psfrompk1[1])
#az.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/iM000psfrompk2[1])
#az.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/iM000psfrompk3[1])
#az.set_xscale('log')
#az.set_ylim(1.,1.0.3)
#az.grid(True)
```



Caption : The ratio of power spectra for each model at different redshifts calculated in two different ways. The numerator is obtained from the z = 0 power spectrum by multiplying by the square of the scale independent growth function plotted above. The denominator is obtained from CAMB by evolving a set of coupled equations obtaining scale dependent growth with neutrinos sourcing the growth at low k, but hardly at high k. The little difference (about 2 percent for z = 3 in the LCDM model) at high k and about 0.5 percent

In [173]:
```
fig = plt.figure(figsize = (16,6))
ay = plt.subplot(1,2,2)
az = plt.subplot(1,2,1)
ay.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/M000n1psfrompk1[1])
ay.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/M000n1psfrompk2[1])
ay.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/M000n1psfrompk3[1])
#ay.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/M000n1psfrompk1[1])
#ay.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/M000n1psfrompk2[1])
#ay.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/M000n1psfrompk3[1])
ay.set_xscale('log')
```

```
ay.set_ylim(1.,1.03)
ay.set_xlim(xmin = 0.01)
ay.grid(True)

def gsratio(redshift):
        redshift = np.asarray(redshift)
        gg = M000n1.growth(z= redshift)[0]
        gs = M000.growth(z = redshift) [0]

        return (gg/gs)**2
az.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/M000n1psfrompk1[1]/gsratio(1.0))
az.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/M000n1psfrompk2[1]/gsratio(2.0))
az.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/M000n1psfrompk3[1]/gsratio(3.0))
#az.plot(M000n1psfrompk1[0],M000n1psfromtk10[1]/M000n1psfrompk1[1])
#az.plot(M000n1psfrompk2[0],M000n1psfromtk20[1]/M000n1psfrompk2[1])
#az.plot(M000n1psfrompk2[0],M000n1psfromtk30[1]/M000n1psfrompk3[1])
az.set_xscale('log')
az.set_xlim(xmax =0.01)
az.set_ylim(1.,1.03)
az.grid(True)
```



Caption : The ratio of power spectra for M000n1 at different redshifts calculated in two different ways. The large ratio (> 10 percent) at low k is multiplied by the ratio of the square of growth factors of M000 and M000n1 (which is around 5 percent, shown in a figure), to effectively use the growth function of M000 instead of M000n1. At these high k, this is appropriate as the growth is sourced by both neutrinos and matter.

4 Mass Function Results from Fitting Formulae

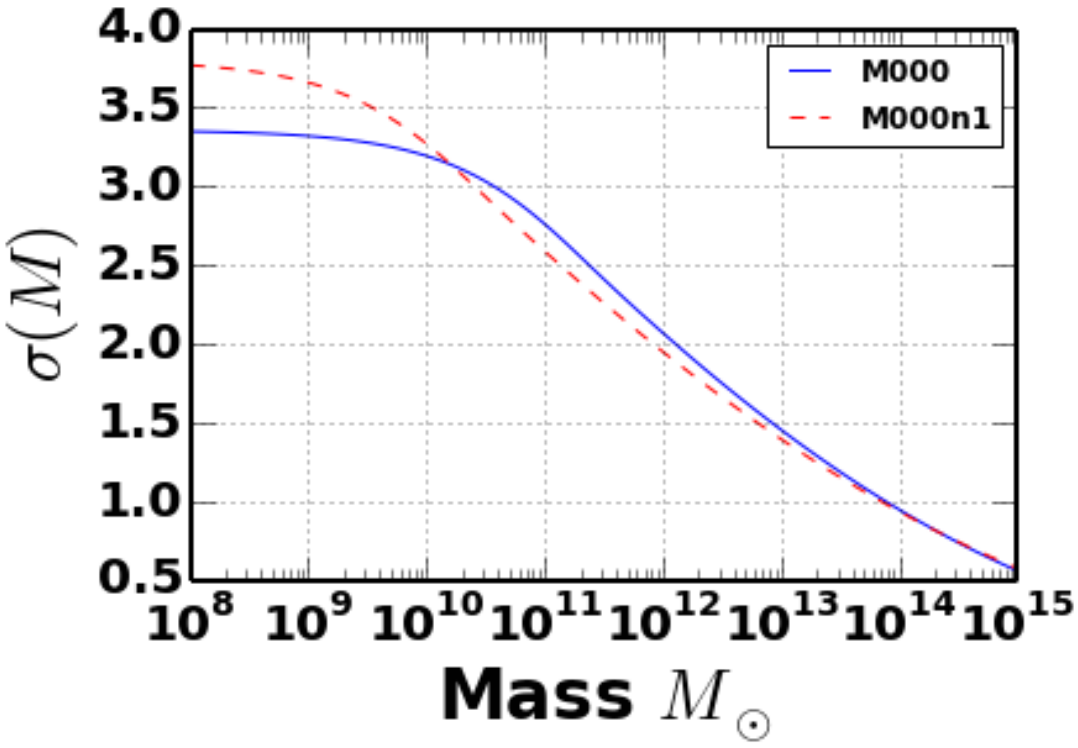
In [204]:
```
Masses = np.logspace(8,15,100)
psfrompkn = psu.powerspectrum(koverh = None, asciifile  = npkfile)#power spectrum outp
psfromtknm = psu.powerspectrum(koverh = None, asciifile  = ntkfile, pstype ="matter",
psfromtkncbs = psu.powerspectrum(koverh = None, asciifile  = ntkfile, pstype ="cb",sig
sigM = psu.sigmaM(M =Masses,ps= psfrompk ,cosmo = M000)
dndlnM_M000 = psu.dndlnM(M =Masses,ps= psfrompk ,cosmo = M000)
dndlnM_M000n1 = psu.dndlnM(M =Masses,ps= psfrompkn ,cosmo = M000n1)
dndlnM_M000n1 = psu.dndlnM(M =Masses,ps= psfrompkn ,cosmo = M000n1)

sigMn = psu.sigmaM(M =Masses,ps= psfrompkn ,cosmo = M000n1)
plt.plot(Masses, sigM,'b-',label = "M000")
plt.plot(Masses, sigMn,'r--',label = "M000n1")
plt.grid(True)
plt.xscale('log')
plt.xlabel("Mass $M_\odot$")
```

```
plt.ylabel("$\sigma (M)$")
plt.legend(loc="best")
```

```
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
```

Out [204]:

```
<matplotlib.legend.Legend at 0x10f93cc10>
```

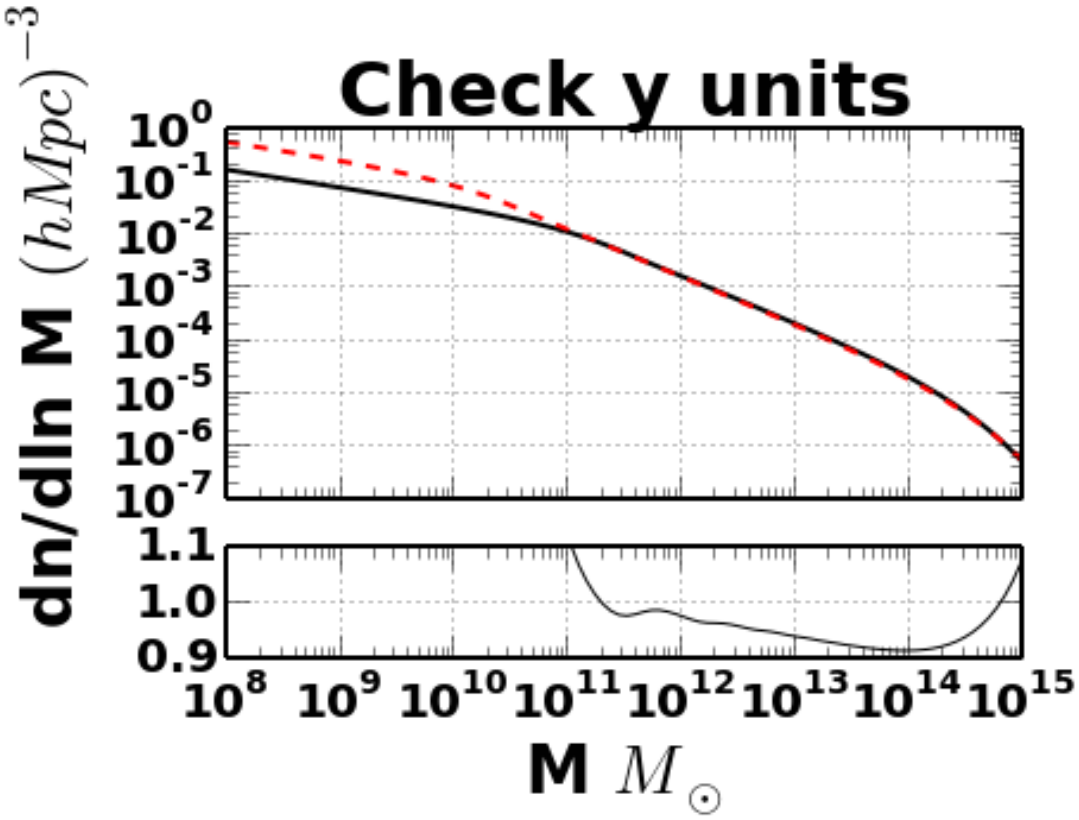


In [216]:

```
mf_fig, mf_ax0, mf_ax1 = pu.settwopanel(setdifflimits=[0.85,1.15])
mf_ax0.loglog(Masses,dndlnM_M000, 'k-',lw=2.0, label = "M000")
mf_ax0.loglog(Masses,dndlnM_M000n1, 'r--',lw =2.0,label = "M000n1")
mf_ax0.xaxis.set_ticklabels("",visible=False)
mf_ax1.plot(Masses, dndlnM_M000n1/dndlnM_M000)
#mf_ax1.yaxis.set_ticks(np.arange(0.9,1.15,0.05))
mf_ax1.set_yticks([0.9,1.0,1.10])
#mf_ax1.yaxis.set_ticklabels([0.9,"",1.0,"",1.10])
mf_ax1.set_ylim(0.9,1.1)
mf_ax1.set_xscale('log')
mf_ax1.set_xlabel("M $M_\odot$")
mf_ax0.set_ylabel("dn/dln M $(h Mpc)^{-3}$")
mf_ax0.set_title("Check y units")
```

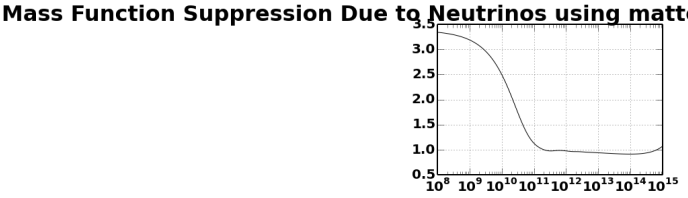
Out [216]:

```
<matplotlib.text.Text at 0x10c886310>
```



In [30]:
```
plt.plot(Masses, dndlnM_M000n1/dndlnM_M000)
plt.xscale('log')
plt.grid(True)
plt.title("Mass Function Suppression Due to Neutrinos using matter power spectrum with
```

Out [30]:
```
<matplotlib.text.Text at 0x10e0a8810>
```



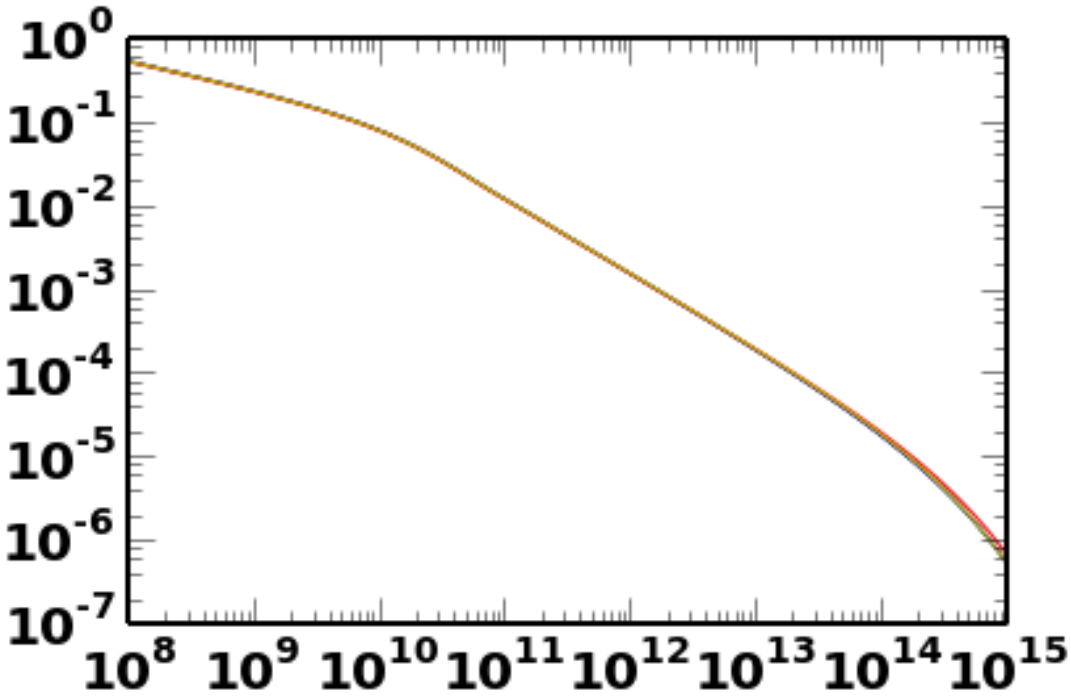
4.1 Mass functions for different neutrino power spectra

In [37]:
```
plt.loglog(Masses, psu.dndlnM(Masses, ps = psfrompkn , cosmo = M000n1))
plt.loglog(Masses, psu.dndlnM(Masses, ps = psfromtknm, cosmo = M000n1))
plt.loglog(Masses, psu.dndlnM(Masses, ps = psfromtkncbs, cosmo = M000n1))
plt.loglog(Masses, psu.dndlnM(Masses, ps = psfromtkncbscb, cosmo = M000n1))
```

```
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
```

Out [37]:

```
[<matplotlib.lines.Line2D at 0x110b14710>]
```



4.2 Mass Functions for Different Normaliztions

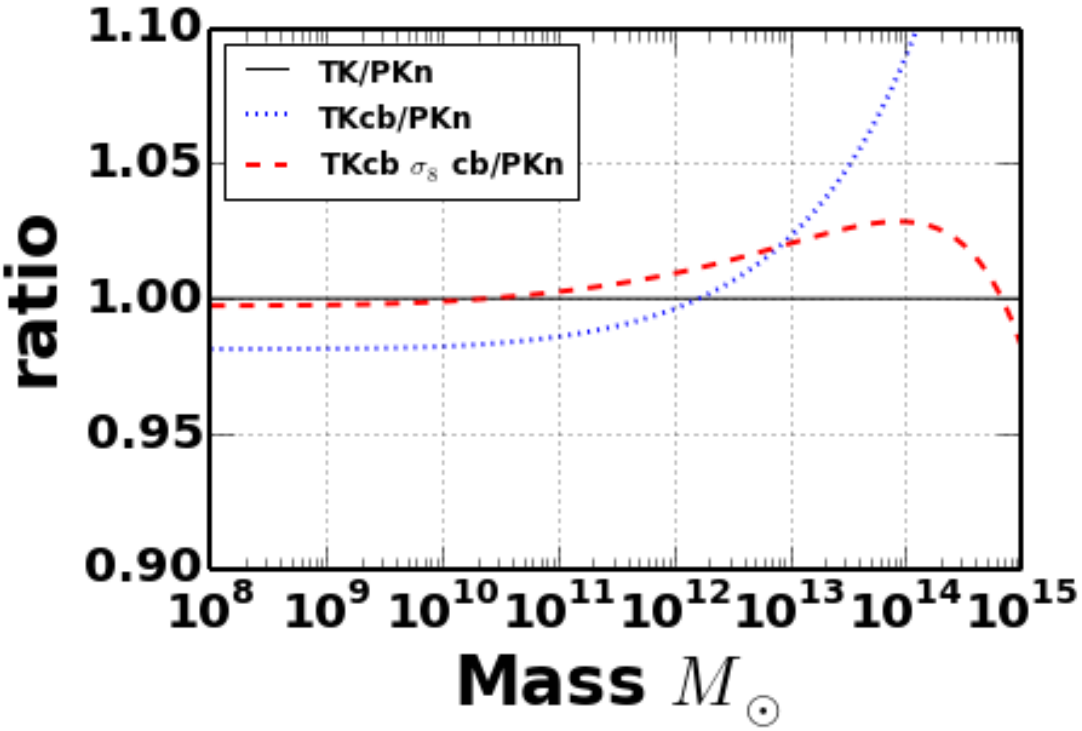
In [74]:

```
#plt.plot(Masses, psu.dndlnM(Masses, ps = psfrompkn , cosmo = M000n1)/psu.dndlnM(Masse
plt.plot(Masses, psu.dndlnM(Masses, ps = psfromtknm, cosmo = M000n1)/psu.dndlnM(Masses
plt.plot(Masses, psu.dndlnM(Masses, ps = psfromtkncbs, cosmo = M000n1)/psu.dndlnM(Mass
plt.plot(Masses, psu.dndlnM(Masses, ps = psfromtkncbscb, cosmo = M000n1)/psu.dndlnM(Ma
plt.xscale('log')
plt.grid(True)
plt.legend(loc= "best")
plt.ylim(0.9,1.1)
plt.xlim(1.0e8,1.0e15)
plt.xlabel(r'Mass $M_\odot$')
plt.ylabel('ratio')
```

```
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
```

Out [74]:

```
<matplotlib.text.Text at 0x110185290>
```



Caption: Ratio of Mass function of M000n1 at z = 0, using power spectrum in different ways. The denominator always uses the CAMB matter power spectrum output of M000n1 and is called Pkn. The black line shows the ratio of mass function when the power spectrum is calculated from the transfer function and the sigma8 value. This should be 1.0 and any deviations are simply due to the sigma8 calculation differing from CAMB. The blue dotted curve uses the cb part of the power spectrum, with the same sigma8 normalization, so each component has the same primordial amplitude as those used in the black curve. The red dashed curve shows the cb power spectrum, normalized so that the cb power spectrum is normalized to sigma8 =0.8. The power spectrum is the same as the matter power spectrum for this model at low k and higher by a factor of (rhom/rhocb)^2 at high k.

4.3 How Does the Mass function of these different types evolve with redshift

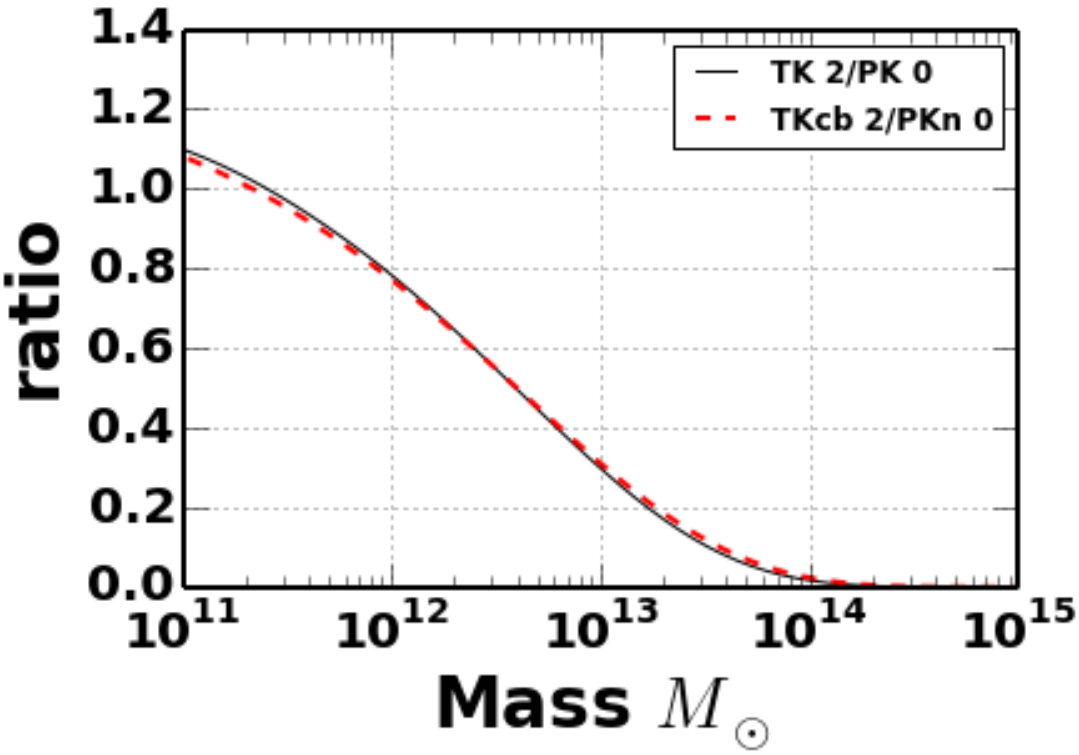
In [223]:

```
#plt.plot(Masses, psu.dndlnM(Masses, ps = psfrompkn , cosmo = M000n1)/psu.dndlnM(Masse
#psfromtkn2  = psu.powerspectrum(koverh = None, asciifile = ntkfile, cosmo = M000n1,z
psfromtk2  = psu.powerspectrum(koverh = None, asciifile = tkfile, cosmo = M000,z =2.0,
psfromtkn2 = psu.powerspectrum(koverh = None,z = 2.0, asciifile  = ntkfile, pstype ="c
plt.plot(Masses, psu.dndlnM(Masses, ps = psfromtk2, cosmo = M000)/psu.dndlnM(Masses, p
plt.plot(Masses, psu.dndlnM(Masses, ps = psfromtkn2, cosmo = M000n1)/psu.dndlnM(Masses

#plt.plot(Masses, psu.dndlnM(Masses, ps = psfromtkncbscb, cosmo = M000n1)/psu.dndlnM(M
plt.xscale('log')
plt.grid(True)
plt.legend(loc= "best")
#plt.ylim(0.8,1.2)
plt.xlim(1.0e11,1.0e15)
plt.xlabel(r'Mass $M_\odot$')
plt.ylabel('ratio')
```

```
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
0.333 0.788 0.807 1.795
```

Out [223]:

```
<matplotlib.text.Text at 0x1113a2f10>
```



Caption: Evolution of the mass function with neutrinos is not tremendously different from the evolution of mass function without neutrinos. The curves show the ratio mass at z = 2.0 (for an exaggerated effect which is still tiny), to the mass function at z = 0.0 for both the M000 model and M000n1 model. For the M000n1 model, we use the cb power spectrum normalized so that the cb sigma8 = 0.8

In []: