HaloMassFunction

December 20, 2023

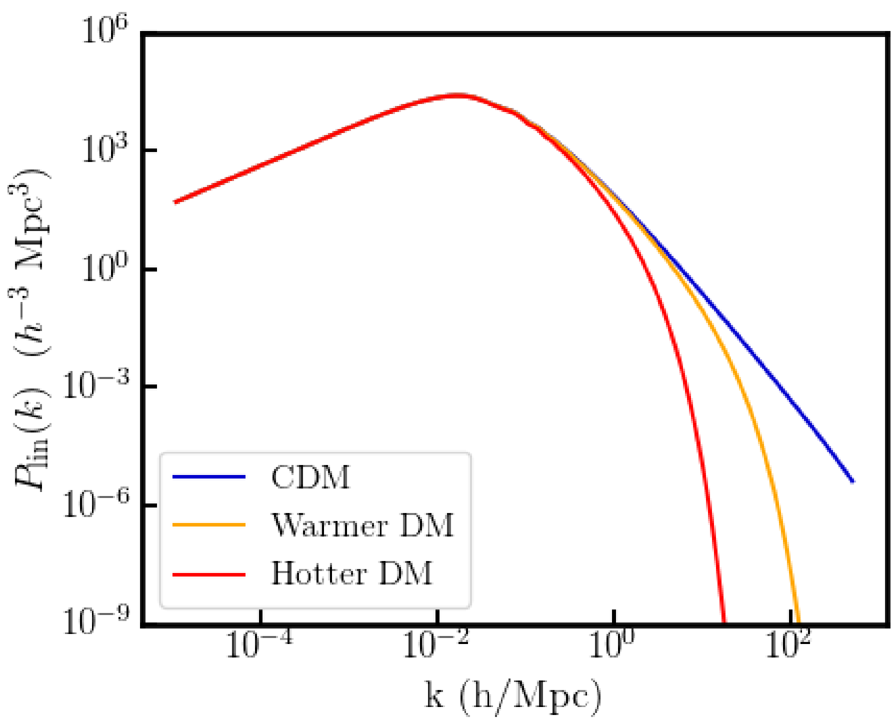
[1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import pandas as pd
from scipy.integrate import quad
from scipy.interpolate import UnivariateSpline, interp1d
import warnings;warnings.filterwarnings('ignore')
from tqdm import tqdm
import astropy.units as u
from astropy.constants import G

plt.rc('font', family = 'serif')
plt.rc('axes', lw = 2); plt.rc('text',usetex = True)
```

[2]:
```
OmegaM0 = 0.3175,
OmegaR0 = 8.2 * 10**(-5)
h = 0.6711
H0 = 100*h*u.km/u.s/u.Mpc
rhobarM = 3*H0**2 / (8*np.pi*G)
rhobarM_h2MsunMpc3 = (rhobarM.to(u.Msun/u.Mpc**3) / h**2).value
```

[3]:
```
df = pd.read_csv('./linear_pk.txt', delim_whitespace = True, header = None,␣
↪skiprows = 7)
df.columns = ['k','Pk']
interpolatorPkCDM = interp1d(df.k,df.Pk)
interpolatorPkWarm = interp1d(df.k,df.Pk * np.exp(-0.1*(df.k)))
interpolatorPkHot = interp1d(df.k,df.Pk * np.exp(-(df.k)))
```

[4]:
```
plt.figure(figsize = (5,4))
ax = plt.gca()
plt.plot(df.k,df.Pk, color = 'mediumblue', label = 'CDM')
plt.plot(df.k,interpolatorPkWarm(df.k), color = 'orange', label = 'Warmer DM')
plt.plot(df.k,interpolatorPkHot(df.k), color = 'red', label = 'Hotter DM')
plt.xscale('log'); plt.yscale('log'); plt.ylim(1e-9,1e6)
plt.ylabel(r'$P_{\rm lin}(k)$ \ ($h^{-3} \rm \ Mpc^3$) ', fontsize = 15); plt.
↪xlabel('k (h/Mpc)', fontsize = 15)
```

```
ax.tick_params(which = 'major', direction = 'in', length = 6, width = 1.5,␣
↪labelsize = 14)
ax.minorticks_on()
ax.tick_params(which = 'minor', direction = 'in', length = 3, width = 1.25,␣
↪labelsize = 14)
plt.legend(loc = 'lower left', fontsize = 13)
plt.savefig('PkComparison.pdf',bbox_inches = 'tight')
```



[5]:
```
def tophat_filter(k, R):
    wtilde = 3 * (k*R)**-3 * (np.sin(k*R) - (k*R) * np.cos(k*R))
    return wtilde
```

[6]:
```
def mass_variance_integrand(k,R, interpfunc):
    wtilde_squared = tophat_filter(k,R)**2
    integrand = (1/(2*np.pi**2)) * interpfunc(k) * wtilde_squared * k**2
    return integrand
```

[7]:
```
# not relevant in report because I assume z = 0.
growth_function_D = UnivariateSpline([0,0.5,1,2],[1, 0.76872625,0.60653086,0.
↪41693915])
```

0.0.1 Set up functions for main calculations + some helper functions

[8]:
```
def convert_M_to_R(M):
        R = (M / ((4/3 * np.pi * rhobarM_h2MsunMpc3)))**(1/3)
        return R
```

[9]:
```
def convert_R_to_M(R):
        M = (4/3) * rhobarM_h2MsunMpc3 * np.pi * R**3
        return M
```

[10]:
```
def mass_variance(M, interpfunc):
    RList = convert_M_to_R(M)
    k_i = 0
    k_f = 100
    sigmaM = np.asarray([np.sqrt(quad(mass_variance_integrand,k_i,k_f, args =␣
 ↪(Rval,interpfunc))[0]) for Rval in RList])

    sigma8 =np.asarray([np.sqrt(quad(mass_variance_integrand,k_i,k_f, args =␣
 ↪(8*h,interpfunc))[0]) for Rval in RList])

    return sigmaM / sigma8
```

[11]:
```
def delta_c(z):
    return 1.686/growth_function_D(z) ## returns a scalar value

def nu(z,Mlist,interpfunc):
    return delta_c(z) / mass_variance(Mlist,interpfunc) ## returns a list

def fps(z,Mlist,interpfunc):
    return np.sqrt(2/np.pi) * nu(z,Mlist,interpfunc) * np.exp(-0.5 *␣
 ↪nu(z,Mlist,interpfunc)**2)

def mass_function(z,Mlist,interpfunc):

    logMasses = np.log(Mlist)
    log_nu_vals = np.log(nu(z,Mlist,interpfunc))
    nu_deriv = UnivariateSpline(np.log(Mlist),log_nu_vals).derivative()

    return rhobarM_h2MsunMpc3/Mlist**2 * fps(z,Mlist,interpfunc) * np.
 ↪gradient(log_nu_vals, logMasses)  * Mlist
```

[12]:
```
## Run Calculations
masslist = np.logspace(8,16,40)
nMz0_cdm = mass_function(0,masslist,interpolatorPkCDM )
nMz0_wdm = mass_function(0,masslist,interpolatorPkWarm )
nMz0_hdm = mass_function(0,masslist,interpolatorPkHot)
```

[13]:

```
plt.figure(figsize = (4,4))
plt.plot(masslist[2:],nMz0_cdm[2:], label = 'z=0; CDM', color = 'mediumblue') #␣
↪ignore first two elements because derivative is poorly defined
plt.plot(masslist[2:],nMz0_wdm[2:], label = 'z=0; Warmer DM', color = 'orange')
plt.plot(masslist[2:],nMz0_hdm[2:], label = 'z=0; Hot DM', color = 'red')
ax = plt.gca()
plt.xscale('log'); plt.yscale('log')
ax.tick_params(which = 'major', direction = 'in', length = 6, width = 1.5,␣
↪labelsize = 14)
ax.minorticks_on()
ax.tick_params(which = 'minor', direction = 'in', length = 3, width = 1.25,␣
↪labelsize = 14)
plt.legend(loc = 'lower left', fontsize = 13)
plt.legend(loc = 'lower left')
plt.ylabel(r'$\frac{dN}{d \ln M} \ (h^{3} \ \rm Mpc^{-3})$', fontsize = 15)
plt.xlabel(r'M ($M_{\odot}$)', fontsize = 15)

ax.set_xticks([10**8, 10**10,10**12, 10**14, 10**16], y = -0.05)

plt.savefig('HMF_Comparison.pdf',bbox_inches = 'tight')
```