testHOD

August 29, 2023

[1]:
```
import os,sys
sys.path.append('./hmvec-master/')
import hmvec as hm # Git clone and pip install as in readme from github.com/
↪msyriac/hmvec
from compute_power_spectra import *
from plotting import *
from params import *

np_load_old = np.load
np.load     = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)
```

[2]:
```
from scipy.interpolate import interp2d,interp1d
import scipy.interpolate as si
```

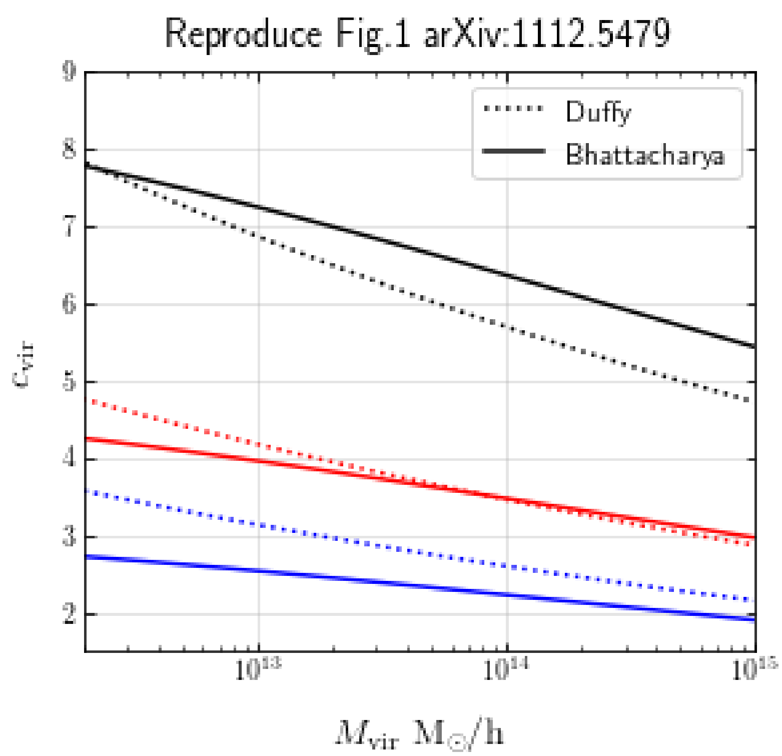
[3]:
```
ellMax = 10000
ells = np.arange(ellMax)

getgas = True
dictKey = dictKey_gas
model = modelParams_gas
rscale = False

cych = ['#377eb8', '#ff7f00', 'forestgreen', '#f781bf', '#a65628', '#984ea3',␣
↪'#999999', '#e41a1c', '#dede00']

baseline = ghztoev(30)

ztype = [6.]
zreio = 6.
nZs = 50

compute_noise = False
compute_BB_noise = False

fsky = [0.7, 0.5, 0.5]
```

[4]:
```
hlil = 0.673
```

```
[5]: if True:
    zMin = 0.005
    zMax = 4.
    nZs = 50

    mMin = 7e8/hlil
    mMax = 3.5e15/hlil
    ms  = np.geomspace(mMin,mMax,100)       # masses
    zs  = np.geomspace(zMin,zMax,nZs)        # redshifts
    ks  = np.geomspace(1e-4,1e3,1001)       # wavenumbers

    # Halo Model
    hcos = hm.HaloModel(zs, ks, ms=ms, mass_function='tinker', mdef='vir')
    #gas = hcos.add_battaglia_profile("y", family="AGN", xmax=2, nxs=30000)

    hod_name = "unWISE blue"
    hcos.add_hod(name=hod_name)

    chis     = hcos.comoving_radial_distance(zs)
    rvirs    = hcos.rvir(ms[None,:],zs[:,None])
    cs       = hcos.concentration()
    Hz       = hcos.h_of_z(zs)
    nzm      = hcos.get_nzm()
    biases   = hcos.get_bh()
    deltav   = hcos.deltav(zs)
    rhocritz = hcos.rho_critical_z(zs)
    dvols    = get_volume_conv(chis, Hz)
    ms200, rs200, cs200 = hcos.mrc200()
```

```
[6]: cs1 = hcos.concentration(mode='duffy')
cs2 = hcos.concentration(mode='BHATTACHARYA')

a0 = np.argmin(np.abs(zs-0.))
a1 = np.argmin(np.abs(zs-1.))
a2 = np.argmin(np.abs(zs-2.))

fig, ax = plt.subplots(1, 1, figsize=(4.5, 4))

plt.plot(ms*hlil, cs1[a0,:], color='k', ls=':', label='Duffy')
plt.plot(ms*hlil, cs2[a0,:], color='k', ls='-', label='Bhattacharya')
#plt.plot(ms*hlil, cs200[a0,:], color='k', ls='--', label='Bhattacharya 200')

plt.plot(ms*hlil, cs1[a1,:], color='r', ls=':')
plt.plot(ms*hlil, cs2[a1,:], color='r', ls='-')
#plt.plot(ms*hlil, cs200[a1,:], color='r', ls='--')

plt.plot(ms*hlil, cs1[a2,:], color='b', ls=':')
```

```
plt.plot(ms*hlil, cs2[a2,:], color='b', ls='-')
#plt.plot(ms*hlil, cs200[a2,:], color='b', ls='--')

plt.xscale('log')
plt.xlim(2.e12, 1e15)
plt.ylim(1.5, 9)
plt.legend()
ax.set_xlabel(r'$M_{\rm vir} {\rm \; M_\odot/h}$'); ax.set_ylabel(r'$c_{\rm␣
↪vir}$')
ax.set_title('Reproduce Fig.1 arXiv:1112.5479')
ax.yaxis.set_ticks_position('both'); ax.xaxis.set_ticks_position('both')
ax.tick_params(which='both', axis="y", direction="in"); ax.
↪tick_params(which='both', axis="x", direction="in")
plt.grid(alpha=0.4)
plt.show()
```

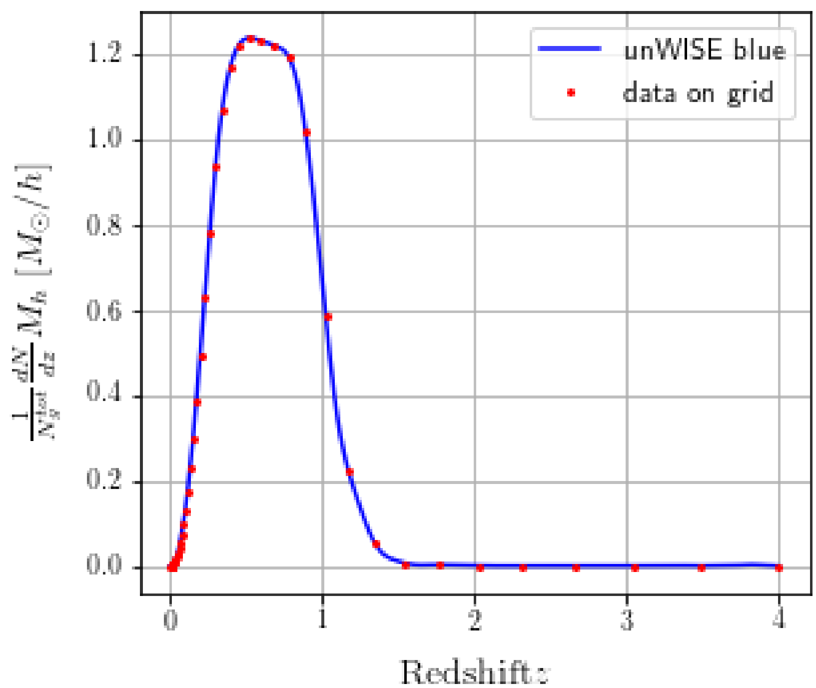


[7]:
```
dndz_data = np.transpose(np.loadtxt("./hmvec-master/data/blue.txt",␣
↪dtype=float))
dndz = np.interp(zs, dndz_data[0,:], dndz_data[1,:])
N_gtot = np.trapz(dndz, zs, axis=0)
W_g = dndz/N_gtot
dndz_data[1,:] = dndz_data[1,:]/N_gtot
```

```
print(np.trapz(W_g, zs))
```

```
1.0
```

[8]:
```
fig, ax = plt.subplots(1, 1, figsize=(4.5, 4))
plt.plot(dndz_data[0,:], dndz_data[1,:], color='b', label='unWISE blue')
plt.plot(zs, W_g, color='r', marker='o', ls='None', ms=2, label='data on grid')

#a = [round(xi, 1) for xi in np.linspace(0, 1.2, 5)]
#ax.set_yticks(a)
#a = [al for aind, al in enumerate(a)]
#ax.set_yticklabels(a)

plt.ylabel(r'$\frac{1}{N^{tot}_g} \frac{d N}{d z} M_h {\; [M_\odot/h]}$')
plt.xlabel(r'${\rm Redshift} z$')
plt.grid()
plt.legend()
plt.show()
```

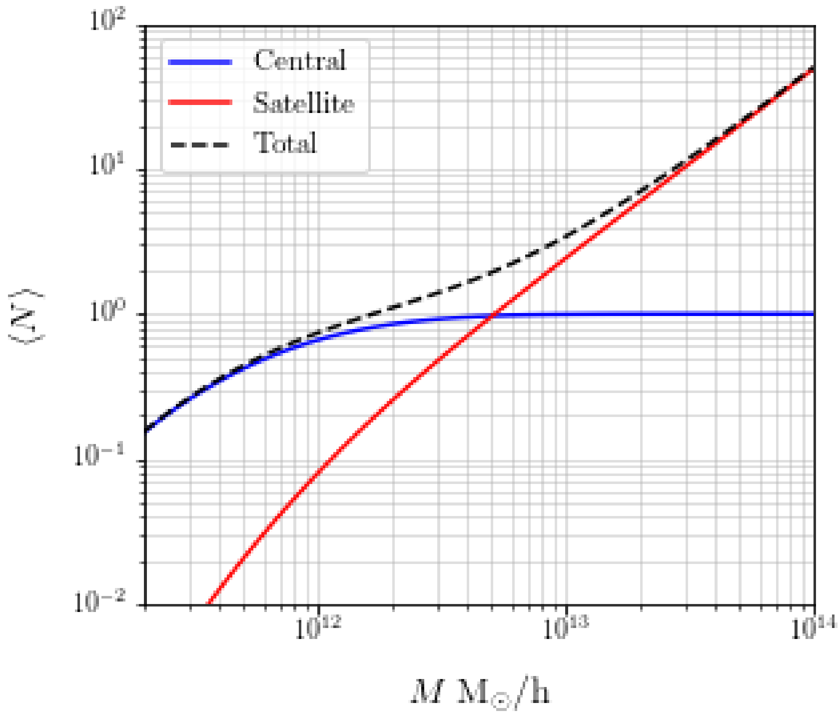


[9]:
```
Ncs = hcos.hods[hod_name]['Nc']
Nss = hcos.hods[hod_name]['Ns']
ngal = hcos.hods[hod_name]['ngal']
bgal = hcos.hods[hod_name]['bg']
```

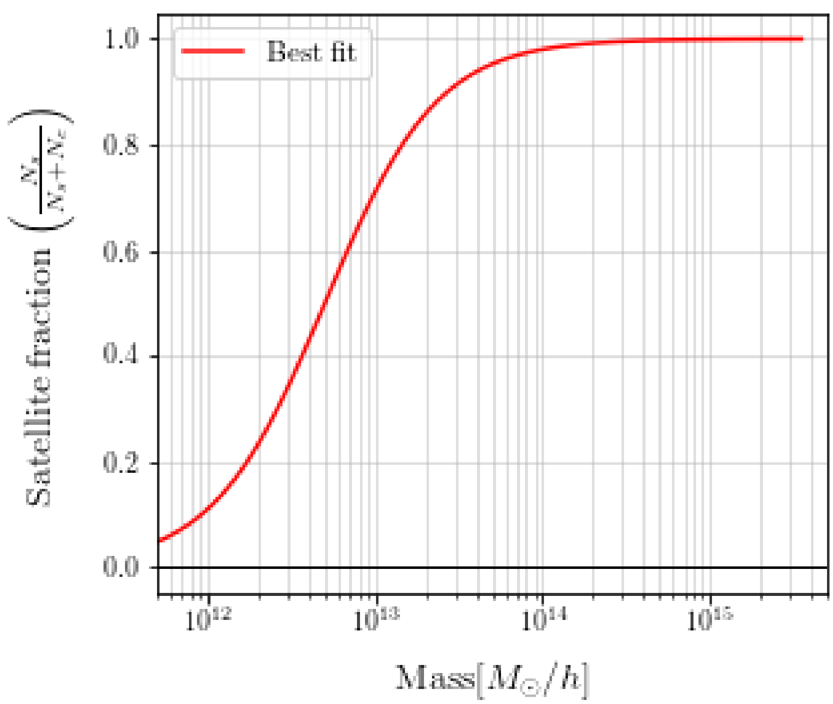
```
satellite_profile_name = hcos.hods[hod_name]['satellite_profile']
print(satellite_profile_name)
central_profile_name = hcos.hods[hod_name]['central_profile']
print(central_profile_name)
```

```
nfw
None
```

[10]:
```
fig, ax = plt.subplots(1, 1, figsize=(4.5, 4))
plt.plot(ms*hlil, Ncs[a0,:], ls='-', color='b', label=r'$\rm Central$')
plt.plot(ms*hlil, Nss[a0,:], ls='-', color='r', label=r'$\rm Satellite$')
plt.plot(ms*hlil, (Ncs+Nss)[a0,:], ls='--', color='k', label=r'$\rm Total$')

plt.ylim(1e-2, 1e2)
plt.xlim(2e11, 1e14)
plt.xlabel(r'$M {\rm \; M_\odot/h}$')
plt.ylabel(r'$\left< N \right>$')
plt.xscale('log')
plt.yscale('log')
plt.grid(True, which="both", ls="-", alpha=0.5)
plt.legend()
plt.show()
```



[11]:
```python
# Reproduce Fig. 10 satellite fraction per halo
fig, ax = plt.subplots(1, 1, figsize=(4.5, 4))
plt.plot(ms*hlil, (Nss/(Ncs+Nss))[a0,:], ls='-', color='r', label=r'$\rm Best \;
 ↪ fit$')

plt.ylim(-0.05, 1.05)
plt.xlim(5e11, 5e15)
plt.ylabel(r'${\rm Satellite \; fraction} \left( \frac{N_s}{N_s + N_c}␣
 ↪\right)$')
plt.xlabel(r'${\rm Mass} [M_\odot/h]$')
plt.xscale('log')
plt.grid(True, which="both", ls="-", alpha=0.5)
plt.axhline(0, color='k', linewidth=1)
plt.legend()
plt.show()
```



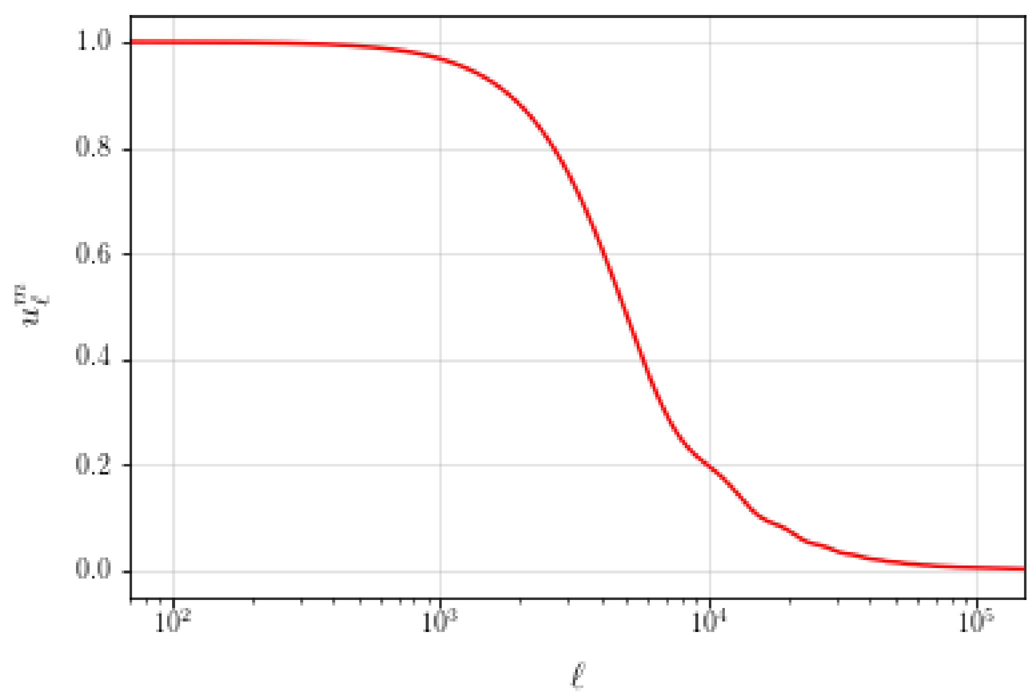
[12]:
```python
# uc = 1 means central galaxies sit at the centres of halos; no spatial distrib
# us = NFW(k) satellites follow NFW profile; this one is projected into Fourier␣
 ↪modes

hod, uc, us = hcos._get_hod_common(hod_name)
print(uc)
print(np.shape(us))
```

```
1.0
(50, 100, 1001)
```

[13]:

```python
# Reproduce Fig. 15: projection of NFW

# get indices of z,m,chi that corresp to params in image description
aid = np.argmin(np.abs(zs-1))
mid = np.argmin(np.abs(ms-3e14/hlil))
cid = np.argmin(np.abs(chis-1317/hlil))

# check that the concentration matches description
print(cs[aid,mid], cs200[aid,mid])

ellks = ks * chis[cid] - 0.5
plt.plot(ellks, us[a0,mid], 'r')

plt.grid(alpha=0.4)
plt.xlim((70,1.5e5))
plt.xscale('log')
plt.xlabel(r'$\ell$')
plt.ylabel(r'$u_\ell^m$')
plt.savefig("./plots/uell.pdf")
plt.show()
```

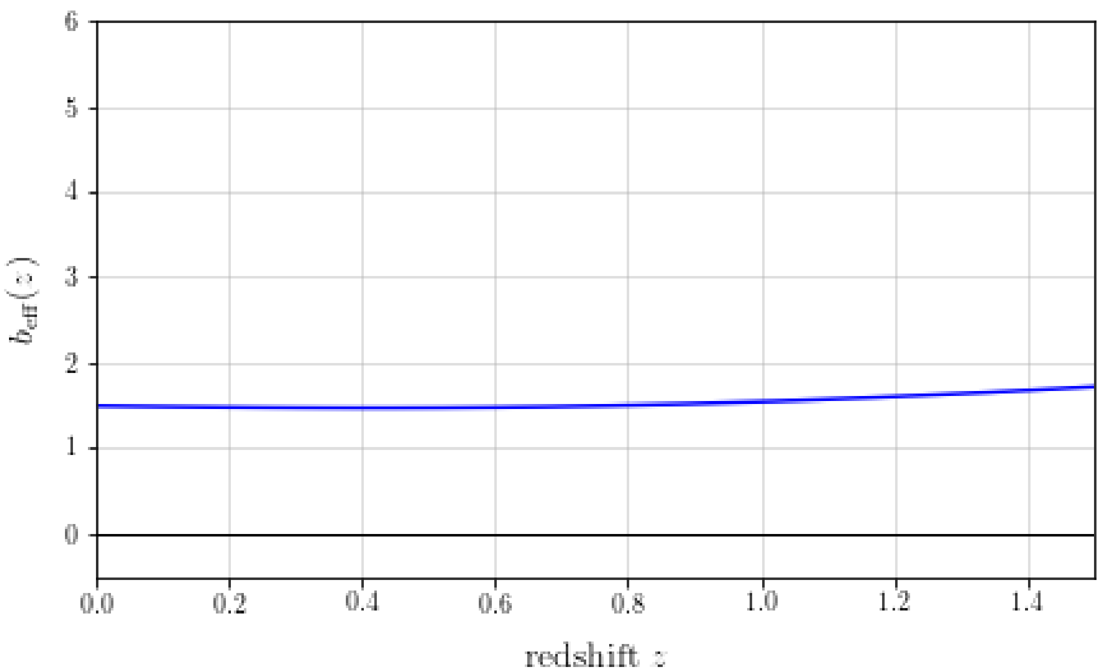
```
3.248584379528022 2.9139311434652866
```



[14]:

```
# Reproduce Fig. 11 galaxy bias
beff = 1./ngal * np.trapz(nzm * biases * (Ncs + Nss), ms, axis=1)
bg = np.trapz(W_g*beff, zs, axis=0)

print(np.count_nonzero((np.round(bgal,10)-np.round(beff,10)).flatten()))
print('Best fit mean galaxy bias:', bg, 'cf. paper value b_g = 1.49')

fig, ax = plt.subplots(1, 1, figsize=(7, 4))
plt.plot(zs, beff, 'b')
plt.ylabel(r'$b_{\rm eff}(z)$')
plt.xlabel(r'${\rm redshift \;} z$')
plt.xlim((0, 1.5))
plt.ylim((-0.5, 6))
plt.grid(alpha=0.5)
plt.axhline(0, color='k', linewidth=1)
plt.show()
```

```
0
Best fit mean galaxy bias: 1.4984166581155305 cf. paper value b_g = 1.49
```

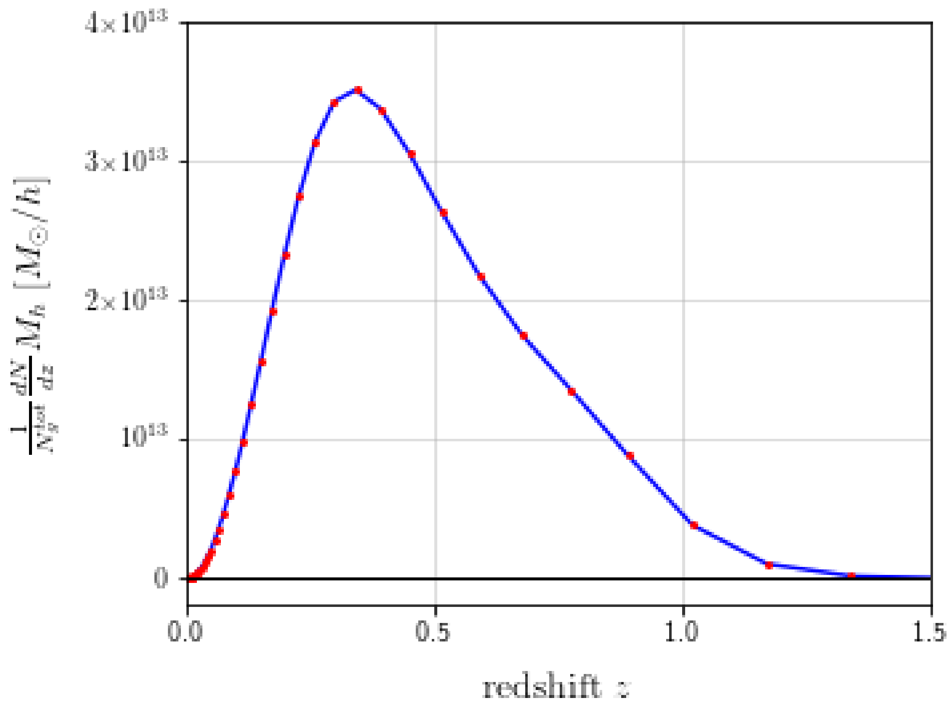


[15]:

```
# Reproduce Fig. 12 mean host halo mass
# This one's off by a factor of 4??
```

```python
massh = W_g/ngal * np.trapz(nzm * ms*hlil * (Ncs + Nss), ms*hlil, axis=1)
avmh = np.trapz(massh, zs, axis=0)

print('Best fit halo mass:', avmh/1e13, 'cf. paper value Mh = 1.99 Msolar/h')

fig, ax = plt.subplots(1, 1, figsize=(5, 4))
plt.plot(zs, massh, 'b')
plt.plot(zs, massh, 'ro', ms=2)
plt.ylabel(r'$\frac{1}{N^{tot}_g} \frac{d N}{d z} M_h {\; [M_\odot/h]}$')
plt.xlabel(r'${\rm redshift \;} z$')

a = [round(xi, 1) for xi in np.linspace(0, 1.5, 4)]
ax.set_xticks(a)
a = [al for aind, al in enumerate(a)]
ax.set_xticklabels(a)

a = ax.get_yticks()[1::2]
ax.set_yticks(a)
a = [fmt(al) for aind, al in enumerate(a)]
ax.set_yticklabels(a)

plt.xlim((0, 1.5))
#plt.ylim((-1e12, 3.9e13))
plt.grid(alpha=0.5)
plt.axhline(0, color='k', linewidth=1)
plt.show()
```

```
Best fit halo mass: 1.9385119341692065 cf. paper value Mh = 1.99 Msolar/h
```



[16]:
```
# Reproduce alpha sat
alphsat = np.trapz(W_g/ngal * np.trapz(nzm * Nss, ms, axis=1), zs, axis=0)

print('Best fit halo mass:', alphsat, 'cf. best fit paper value a_sat = 0.30')
```

```
Best fit halo mass: 0.33713225849177947 cf. best fit paper value a_sat = 0.30
```

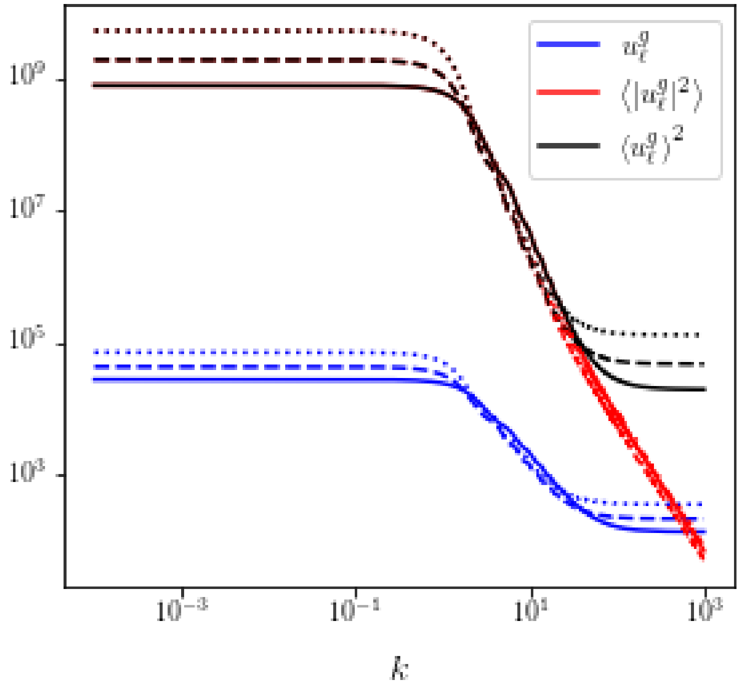
[17]:
```
# HOD

uk_g = (Ncs[...,None] + us * Nss[...,None]) / ngal[:,None,None]
uk_gsq = (2. * us * Nss[...,None] + us**2. * Nss[...,None]**2.) / ngal[:
↪,None,None]**2.
```

Interpolate NFW profiles uk, uksq and lin mat. pow. Pzk onto ells: ks = (ell+0.5)/chis

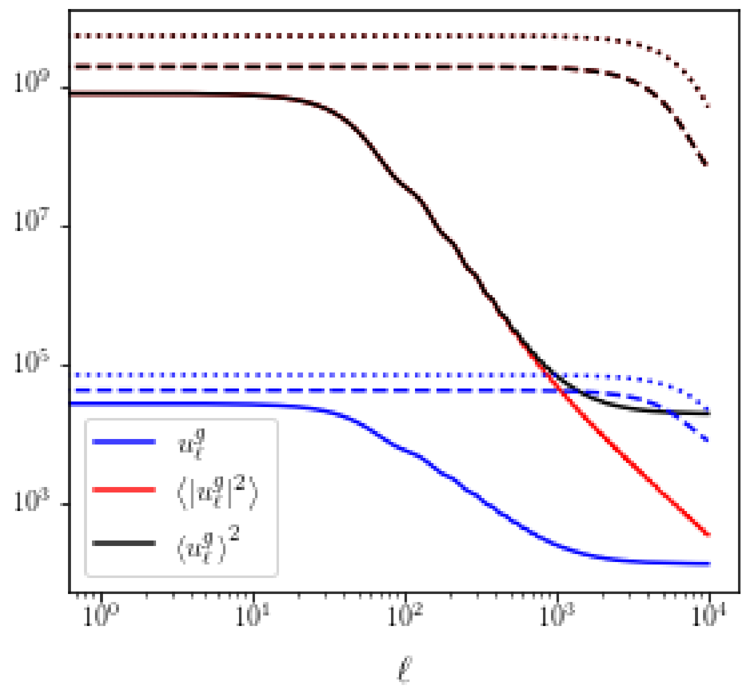
[18]:
```
uell_profile, uellsq_profile = np.zeros((2, len(zs), len(ms), len(ells)))
Pzell = np.zeros((len(zs), len(ells)))

f = interp2d(ks, zs, hcos.Pzk, bounds_error=True)
for ii, ell in enumerate(ells):
    kevals = (ell+0.5)/chis
    interpolated = si.dfitpack.bispeu(f.tck[0], f.tck[1], f.tck[2], f.tck[3], f.
↪tck[4], kevals, zs)[0]
    Pzell[:, ii] = interpolated
```

```
for mi, mm in enumerate(ms):
    f = interp2d(ks, zs, uk_g[:,mi,:], bounds_error=True)
    for ii, ell in enumerate(ells):
        kevals = (ell+0.5)/chis
        interpolated = si.dfitpack.bispeu(f.tck[0], f.tck[1], f.tck[2], f.
↪tck[3], f.tck[4], kevals, zs)[0]
        uell_profile[:, mi, ii] = interpolated

    f = interp2d(ks, zs, uk_gsq[:,mi,:], bounds_error=True)
    for ii, ell in enumerate(ells):
        kevals = (ell+0.5)/chis
        interpolated = si.dfitpack.bispeu(f.tck[0], f.tck[1], f.tck[2], f.
↪tck[3], f.tck[4], kevals, zs)[0]
        uellsq_profile[:, mi, ii] = interpolated
```

[19]:
```
fig, ax = plt.subplots(1, 1, figsize=(4.5, 4))
plt.plot(ks, uk_g[a0,mid,:], ls='-', color='b', label=r'$u^g_\ell$')
plt.plot(ks, uk_gsq[a0,mid,:], ls='-', color='r', label=r'$\left< |u^g_\ell|^2␣
↪\right>$')
plt.plot(ks, uk_g[a0,mid,:]**2, ls='-', color='k', label=r'$\left< u^g_\ell␣
↪\right>^2$')

plt.plot(ks, uk_g[a1,mid,:], ls='--', color='b')
plt.plot(ks, uk_gsq[a1,mid,:], ls='--', color='r')
plt.plot(ks, uk_g[a1,mid,:]**2, ls='--', color='k')

plt.plot(ks, uk_g[a2,mid,:], ls=':', color='b')
plt.plot(ks, uk_gsq[a2,mid,:], ls=':', color='r')
plt.plot(ks, uk_g[a2,mid,:]**2, ls=':', color='k')

plt.xlabel(r'$k$')
plt.xscale('log')
plt.yscale('log')

plt.legend()
plt.show()
```



[20]:
```
fig, ax = plt.subplots(1, 1, figsize=(4.5, 4))
plt.plot(ells, uell_profile[a0,mid,:], ls='-', color='b', label=r'$u^g_\ell$')
plt.plot(ells, uellsq_profile[a0,mid,:], ls='-', color='r', label=r'$\left<␣
 ↪|u^g_\ell|^2 \right>$')
plt.plot(ells, uell_profile[a0,mid,:]**2, ls='-', color='k', label=r'$\left<␣
 ↪u^g_\ell \right>^2$')

plt.plot(ells, uell_profile[a1,mid,:], ls='--', color='b')
plt.plot(ells, uellsq_profile[a1,mid,:], ls='--', color='r')
plt.plot(ells, uell_profile[a1,mid,:]**2, ls='--', color='k')

plt.plot(ells, uell_profile[a2,mid,:], ls=':', color='b')
plt.plot(ells, uellsq_profile[a2,mid,:], ls=':', color='r')
plt.plot(ells, uell_profile[a2,mid,:]**2, ls=':', color='k')

plt.xlabel(r'$\ell$')
plt.xscale('log')
plt.yscale('log')

plt.legend()
plt.show()
```

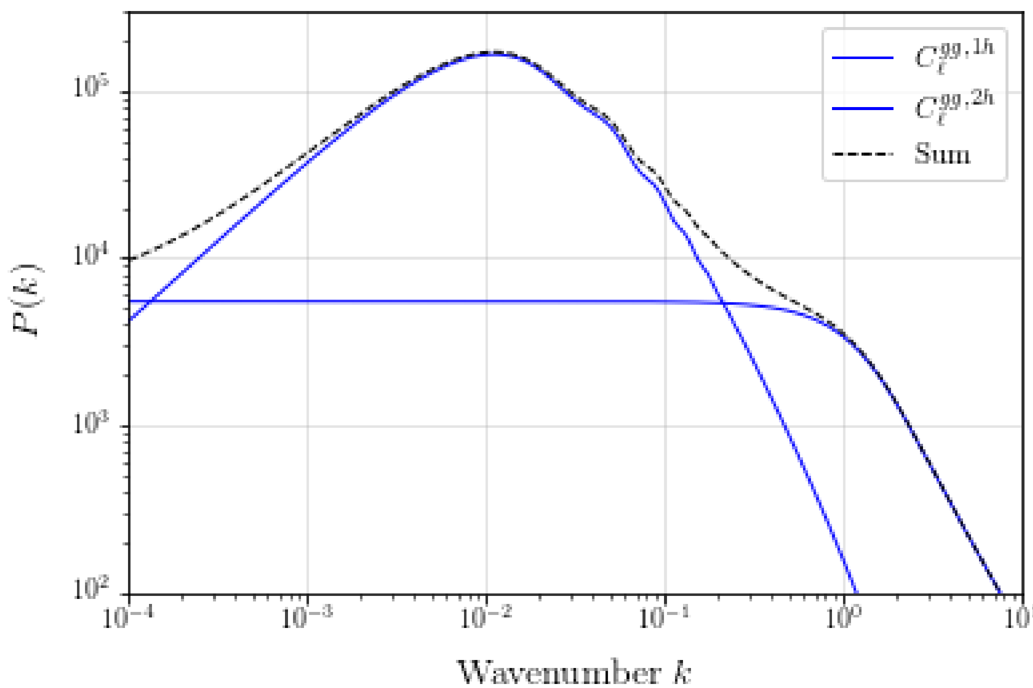


[21]:
```
# Construct power spectra

Cell_1h = (W_g[:,None]/dvols[:,None])**2. * np.trapz(nzm[...,None] *␣
 ↪uellsq_profile, ms, axis=1)
Cell_1h = np.trapz(dvols[:,None] * Cell_1h, zs, axis=0)

intzell = W_g[:,None]/dvols[:,None] * np.trapz(nzm[...,None] * biases[...,None]␣
 ↪* uell_profile, ms, axis=1)
Cell_2h = np.trapz(dvols[:,None] * np.abs(intzell)**2. * Pzell, zs, axis=0)

Cell_tot = Cell_1h + Cell_2h
```

Next: check if this is equivalent to computing power spectra Pk then doing a Limber integral (like in hmvec)

[22]:
```
# Power spectra

Pk_gg_1h = W_g[:,None]**2. * np.trapz(nzm[...,None] * uk_gsq, ms, axis=1)

intzell  = np.trapz(nzm[...,None] * biases[...,None] * uk_g, ms, axis=1)
Pk_gg_2h = W_g[:,None]**2. * np.abs(intzell)**2. * hcos.Pzk

Pk_gg = Pk_gg_1h + Pk_gg_2h
```

[23]:
```
plt.plot(ks, Pk_gg_1h[cid], ls='-', color='b', linewidth=1,␣
↪label=r'$C_\ell^{gg,1h}$')
plt.plot(ks, Pk_gg_2h[cid], ls='-', color='b', linewidth=1,␣
↪label=r'$C_\ell^{gg,2h}$')
plt.plot(ks, Pk_gg[cid], ls='--', color='k', linewidth=1, label=r'$\rm Sum$')

plt.xlabel(r'${\rm Wavenumber \;} k$')
plt.ylabel(r'$P(k)$')
plt.ylim(1e2, 3e5)
plt.xlim(min(ks), 1e1)
plt.xscale('log')
plt.yscale('log')
plt.axhline(0, color='k', linewidth=1.)
plt.grid(True, alpha=0.4)
plt.legend()
plt.show()
```



[24]:
```
def limber_int(ells,zs,ks,Pzks,hzs,chis):
    hzs = np.array(hzs).reshape(-1)
    chis = np.array(chis).reshape(-1)
    prefactor = hzs / chis**2.

    f = interp2d(ks, zs, Pzks, bounds_error=True)
```

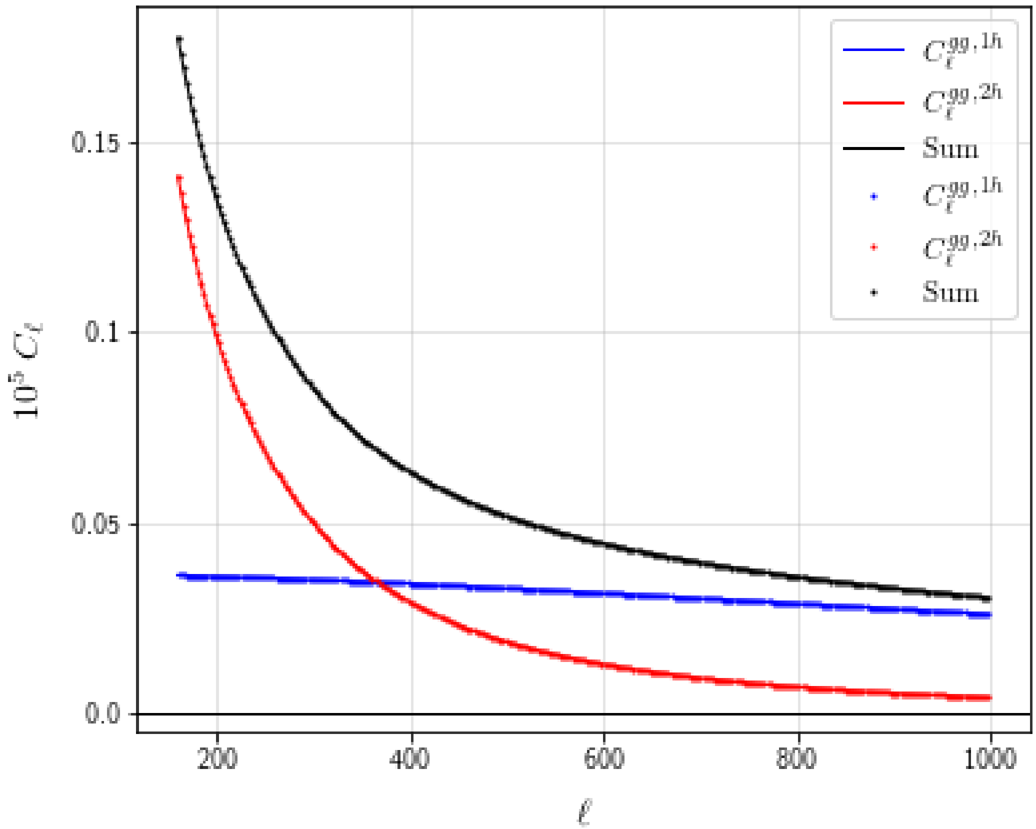
```
    Cells = np.zeros(ells.shape)
    for ii, ell in enumerate(ells):
        kevals = (ell+0.5)/chis
        interpolated = si.dfitpack.bispeu(f.tck[0], f.tck[1], f.tck[2], f.
↪tck[3], f.tck[4], kevals, zs)[0]
        Cells[ii] = np.trapz(interpolated*prefactor, zs)
    return Cells
```

[25]:
```
Cls_gg_1h = limber_int(ells, zs, ks, Pk_gg_1h, Hz, chis)
Cls_gg_2h = limber_int(ells, zs, ks, Pk_gg_2h, Hz, chis)
Cls_gg = limber_int(ells, zs, ks, Pk_gg, Hz, chis)
```

Reproduce Fig. 8 and check that the two metdhos are equivalent

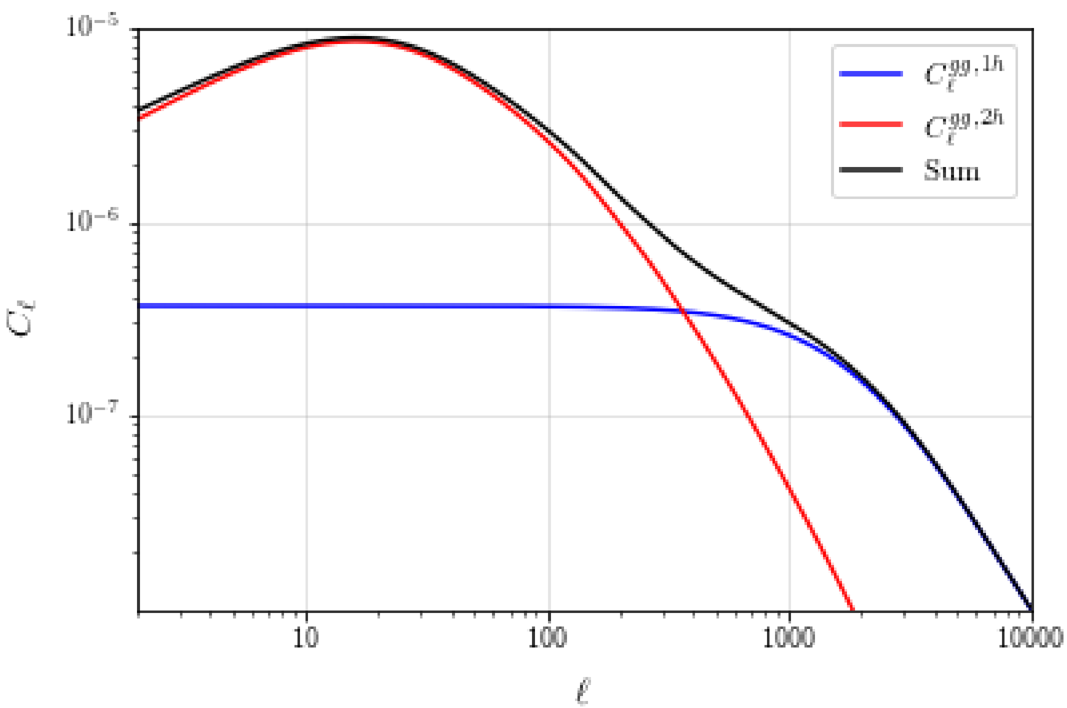
[26]:
```
fig, ax = plt.subplots(1, 1, figsize=(6, 5))

ellr = np.arange(160, 1000)
fact = 1.e5
plt.plot(ells[ellr], fact*Cls_gg_1h[ellr], ls='-', lw=1, color='b',␣
↪label=r'$C_\ell^{gg,1h}$')
plt.plot(ells[ellr], fact*Cls_gg_2h[ellr], ls='-', lw=1, color='r',␣
↪label=r'$C_\ell^{gg,2h}$')
plt.plot(ells[ellr], fact*Cls_gg[ellr], ls='-', lw=1, color='k', label=r'$\rm␣
↪Sum$')

plt.plot(ells[ellr[::3]], fact*Cell_1h[ellr[::3]], 'o', ms=1, color='b',␣
↪label=r'$C_\ell^{gg,1h}$')
plt.plot(ells[ellr[::3]], fact*Cell_2h[ellr[::3]], 'o', ms=1, color='r',␣
↪label=r'$C_\ell^{gg,2h}$')
plt.plot(ells[ellr[::3]], fact*Cell_tot[ellr[::3]], 'o', ms=1, color='k',␣
↪label=r'$\rm Sum$')

a = np.array(np.linspace(200, 1000, 5), dtype='int')
ax.set_xticks(a)
a = [al for aind, al in enumerate(a)]
ax.set_xticklabels(a)

a = [round(xi, 2) for xi in np.linspace(0, 0.15, 4)]
ax.set_yticks(a)
a = [al for aind, al in enumerate(a)]
ax.set_yticklabels(a)

plt.xlabel(r'$\ell$')
plt.ylabel(r'$10^5 \; C_\ell$')
plt.axhline(0, color='k', linewidth=1.)
plt.grid(True, alpha=0.4)
```

```
plt.legend()
plt.show()
```



[27]:
```
""" Reproduce Fig. 14. """

fig, ax = plt.subplots(1, 1, figsize=(6, 4))

ellr = np.arange(2, 10000)
fact = 1.#(ells * (ells+1.)/2./np.pi)[ellr]
plt.plot(ells[ellr], fact*Cls_gg_1h[ellr], ls='-', color='b',␣
 ↪label=r'$C_\ell^{gg,1h}$')
plt.plot(ells[ellr], fact*Cls_gg_2h[ellr], ls='-', color='r',␣
 ↪label=r'$C_\ell^{gg,2h}$')
plt.plot(ells[ellr], fact*Cls_gg[ellr], ls='-', color='k', label=r'$\rm Sum$')

plt.xscale('log')
plt.yscale('log')
plt.ylim((1e-8, 1e-5))
plt.xlim((2, 1e4))
```

```
a = np.array(np.geomspace(1e-7, 1e-5, 3))
ax.set_yticks(a)
a = [fmt(al) for aind, al in enumerate(a)]
ax.set_yticklabels(a)

a = np.array(np.geomspace(10, 1e4, 4), dtype='int')
ax.set_xticks(a)
a = [al for aind, al in enumerate(a)]
ax.set_xticklabels(a)


plt.xlabel(r'$\ell$')
plt.ylabel(r'$C_\ell$')
plt.axhline(0, color='k', linewidth=1.)
plt.grid(True, alpha=0.4)
plt.legend()
plt.show()

# Low ell for 1halo term not great but may be Limber
```




```
[ ]:
```

```
[ ]:
```