

Final Exam: codes

Vladislav Lenskiy

June 15, 2022

For problem 1:

```
1      import sys
2      import astropy
3      from pathlib import Path
4      import camb
5      from camb import model
6      import numpy as np
7      from matplotlib import pyplot as plt
8      from matplotlib import rcParams
9      from astropy import units as u
10     from astropy import constants as c
11
12     from scipy.optimize import curve_fit
13     from scipy.integrate import quad
14
15
16     rcParams.update({'font.size':12})
17     rcParams.update({'text.usetex':True})
18
19     WD = 'D:/SNU/Cosm'
20
21     #Now get matter power spectra and sigma8 at redshift 0 and 0.8
22     npoints = 1000
23     count=0
24     chi=[]
25     As_init = 2.1073e-9
26     h = 0.677
27     omb = 0.048
28     for om in np.arange(0.41,0.43,0.005):
29         print("\nom=",om)
30         chic=[]
31         for sigma in np.arange(0.5,0.55,0.01):
32
33
34             ombh2 = omb * h**2
```

```

35     omch2 = (om-omb) * h**2
36     pars = camb.CAMBparams()
37     pars.set_cosmology(H0=100*h, ombh2=ombh2, omch2=omch2)
38     pars.InitPower.set_params(ns=0.96, As=As_init)
39     pars.set_matter_power(redshifts=[0.], kmax=100.0)
40
41     #Linear spectra
42     pars.NonLinear = model.NonLinear_none
43     results = camb.get_results(pars)
44     kh, z, pk = results.get_matter_power_spectrum(
45     minkh=1e-4, maxkh=10, npoints=npoints)
46     s8 = np.array(results.get_sigma8())
47
48     # considering sigma8 = 0.8228
49     s8_ratio = sigma/s8[0]
50     As = s8_ratio**2 * As_init
51
52     pars = camb.CAMBparams()
53     pars.set_cosmology(H0=100*h, ombh2=ombh2, omch2=omch2)
54     pars.InitPower.set_params(ns=0.96, As=As)
55     pars.set_matter_power(redshifts=[0.], kmax=2.0)
56     #Linear spectra
57     pars.NonLinear = model.NonLinear_none
58     results = camb.get_results(pars)
59     kh, z, pk = results.get_matter_power_spectrum(
60     minkh=1e-4, maxkh=10, npoints=npoints)
61     #Non-Linear spectra
62     pars.NonLinear = model.NonLinear_both
63     results.calc_power_spectra(pars)
64     kh_nonlin, z_nonlin, pk_nonlin =
65     results.get_matter_power_spectrum(
66     minkh=1e-4, maxkh=10, npoints=npoints)
67
68     kh8 = 1 / 8 ## h
69
70

```

```

71         # filters
72     def tophat(x, rf):
73         num = np.sin(x*rf) - x*rf*np.cos(x*rf)
74         den = (x*rf)**3
75         return 3 * num / den
76
77     def gaussian(x, rf):
78         return np.exp(-(x*rf)**2 * 0.5)
79
80     def sharp_k(x, rf):
81         _thres = 1 / rf
82         _filter = np.ones_like(x)
83         _filter[x > _thres] = 0
84         return _filter
85
86     def sig_squared(filt, x, pk, rf):
87         integrand = x**2 * pk *
88         filt(x, rf)**2 / np.pi**2 / 2.
89         return np.trapz(integrand, x)
90
91     def mass_to_r(mass, h, filt='tophat'):
92         _rho = om*2.7754e11 * u.Msun / u.Mpc**3 ## h**2
93
94         gam_f = {'tophat': 4*np.pi/3,
95                  'gaussian': (2*np.pi)**(3/2),
96                  'sharp_k': 6*np.pi**2}
97
98         r3 = mass / _rho / gam_f[filt]
99
100        return r3**(1/3)
101
102    mass = 10*np.linspace(0, 3, npoints) * 1e12*u.Msun
103
104
105
106

```

```

107
108
109     #The Press-Schechter function with factor of 2,
110
111
112
113     rho_mean = om*(3*(100*u.km/u.s/u.Mpc)**2/(8*np.pi*c.G))
114     .to(u.Msun*u.Mpc**(-3))
115     delt_c = 3/5*(3/4)**(2/3)*(2*np.pi)**(2/3)
116
117     func = lambda x : sig_squared(tophat, kh, pk, x)
118     r_f = mass_to_r(mass, 1, filt='tophat')
119     vfunc = np.vectorize(func)
120     sig2 = vfunc(r_f.value)
121     sig = np.sqrt(sig2)
122
123     dlmsig = np.gradient(np.log(sig))
124     dlnM = np.gradient(np.log(mass.value))
125     multi_func_PS = np.sqrt(2/np.pi) * delt_c/sig *
126     np.exp(-0.5*delt_c**2/sig2)
127     dNdlm_PS = multi_func_PS * rho_mean/mass *
128     np.abs(dlmsig/dlnM)
129
130
131     #Sheth-Tormen function
132
133     A = 0.3222
134     a = 0.707
135     P = 0.3
136     nu = delt_c/sig
137     multi_func_ST = A*np.sqrt(2*a/np.pi)*(1+(1/a/nu**2)**P)*
138     nu*np.exp(-a/2*nu**2)
139
140     dNdlm_ST = multi_func_ST * rho_mean/mass *
141     np.abs(dlmsig/dlnM)
142     x_axis=np.log10(mass/u.Msun)

```

```

143         y_axis=dNdlmM_ST
144
145
146
147
148         #Chi-squared!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
149         with open("ST_func_x.csv") as f:
150             emp_x = np.loadtxt(f, delimiter=",")[:,1]
151         with open("ST_func_y.csv") as f:
152             emp_y = np.loadtxt(f, delimiter=",")[:,1]
153         #print(y_axis)
154         #print(emp_x)
155         indexes=[]
156         y_axisu=[]
157         for i in emp_x.round(2):
158             k=0
159             for j in x_axis.round(2):
160                 if i == j:
161                     y_axisu.append(y_axis[k].value)
162                     break
163                 k=k+1
164             #print("\n", "index")
165
166             #print(y_axisu)
167             #print(x_axis[99])
168             chicc=0
169             for i in range(25):
170                 chicc=((y_axisu[i]-emp_y[i])**2)/y_axisu[i]+chicc
171             chic.append(chicc)
172         chi.append(chic)
173     print(chi)
174     min=10
175     indexom=0
176     k=0
177     indexsigma=0
178     for i in chi:

```

```

179         c=0
180         for j in chi[k]:
181             if chi[k][c]<min:
182                 min=chi[k][c]
183                 indexom=k
184                 indexsigma=c
185             c=c+1
186         k=k+1
187     print("\n",indexom)
188     print("\n",indexsigma)
189
190
191     WD = 'D:/SNU/Cosm'
192     fig = plt.figure()
193     ax = fig.add_subplot(111)
194     ax.plot(emp_x ,emp_y, c='k', label='from Rockstar')
195     ax.plot(emp_x, y_axisu, c='b', label='theoretical')
196     plt.title(f'Graph for the best-fit values')
197     ax.set_yscale('log')
198     ax.set_xlabel('$\log M$ (h-1M⊙)')
199     ax.set_ylabel('$dN/d\ln M$ [(h-1Mpc)-3']')
200     plt.legend()
201     plt.tight_layout()
202     plt.savefig(WD+'/FE_N1.png', dpi=300)
203     # def objective(x, a, b):
204     #     return a * x + b
205
206     # # choose the input and output variables
207     # x, y = emp_x, emp_y
208     # # curve fit
209     # popt, _ = curve_fit(objective, x, y)
210     # # summarize the parameter values
211     # a, b = popt
212     # #print('y = %.5f * x + %.5f' % (a, b))
213     # # plot input vs output
214     # plt.plot(x, y)

```

```
215         # # define a sequence of inputs between
216         # #the smallest and largest known inputs
217         # x_line = np.arange(min(x), max(x), 1)
218         # # calculate the output for the range
219         # y_line = objective(x_line, a, b)
220         # # create a line plot for the mapping function
221
222         # y_axisu=[]
223         # for i in emp_x.round(2):
224         #     k=0
225         #     for j in x_axis.round(2):
226         #         if i == j:
227         #             y_axisu.append(y_axis[k].value)
228         #             break
229         #         k=k+1
230         # plt.plot(x, y_axisu)
231         # plt.show()
```

For problem 2:

```
1      import sys, platform, os
2      import matplotlib
3      from matplotlib import pyplot as plt
4      import numpy as np
5      import csv
6      import pandas as pd
7      from astropy import units as u
8      from astropy import constants as c
9
10
11     csvf=pd.read_csv('Rockstar_for_CAMB.csv')
12     mass=csvf["h.Mvir"]
13     # mas=np.asarray(mass)
14     # mas10=np.log10(mas)
15     # mase=np.log(mas)
16     # print(mase[:3])
17     N= []
18     Ma=[]
19     j=0
20     step=10**12
21     M=10**12
22     while M <(10**15)+1 :
23         Ma.append(M)
24         N.append(0)
25         N[j] = sum(map(lambda x : M+step>x>M, mass))
26         j=j+1
27         M=M+step
28         print(j)
29         print('M= ',M)
30         if(j==9 or j==118):
31             step=step*10
32             print('step= ',step)
33         if(j==18 or j==119):
34             step=step*10
```

```

35         print('step= ',step)
36
37
38     mas=np.asarray(Ma)
39     dN=[]
40     dmase=[]
41     dN_dmase=[]
42     for i in range(len(N)-1):
43         dN.append(N[i+1]-N[i])
44         dmase.append(np.log(mas)[i+1]-np.log(mas)[i])
45         dN_dmase.append(dN[i]/dmase[i])
46
47     dN_dmase.pop(17)
48     dN_dmase.pop(8)
49     #dmase.pop(19)
50
51
52
53     # dN=np.gradient(N)
54     # dmase=np.gradient(np.log(mas))
55
56     print('\n',dN)
57     print('\n',dmase)
58     logm=np.log10(mas)
59     # logm=np.delete(logm,-1)
60     logm=np.delete(logm,17)
61     logm=np.delete(logm,8)
62     print('\n',dN_dmase)
63
64     WD = 'D:/SNU/Cosm'
65     fig = plt.figure()
66     ax = fig.add_subplot(111)
67     ax.plot(np.delete(logm,-1) , np.abs(dN_dmase)/(4*(10**8)), c='k')
68     plt.title(f'Using MDPL2.Rockstar data')
69     ax.set_yscale('log')
70     ax.set_xlabel('$\log M$ (h-1M⊙)$')

```

```

71     ax.set_ylabel('$dN/d\ln M$ [$ (h^{-1}Mpc)^{-3}$] ')
72     plt.tight_layout()
73     plt.savefig(WD+'N3_Rockstar.png', dpi=300)
74
75
76     #Chi-squared!!!!!!!!!!!!!!!!!!!!!!
77     pd.DataFrame(np.delete(logm,-1)).to_csv("ST_func_x.csv")
78     pd.DataFrame(np.abs(dN_dmase)/(4*(10**8))).to_csv("ST_func_y.csv")

```