Final Exam: codes

Vladislav Lenskiy

June 15, 2022

For problem 1:

```
import sys
import astropy
from pathlib import Path
import camb
from camb import model
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import rcParams
from astropy import units as u
from astropy import constants as c

from scipy.optimize import curve_fit
from scipy.integrate import quad


rcParams.update({'font.size':12})
rcParams.update({'text.usetex':True})

WD = 'D:/SNU/Cosm'

#Now get matter power spectra and sigma8 at redshift 0 and 0.8
npoints = 1000
count=0
chi=[[]]
As_init = 2.1073e-9
h = 0.677
omb = 0.048
for om in np.arange(0.41,0.43,0.005):
    print("\nom=",om)
    chic=[]
    for sigma in np.arange(0.5,0.55,0.01):


        ombh2 = omb * h**2
```

```
omch2 = (om-omb) * h**2
pars = camb.CAMBparams()
pars.set_cosmology(H0=100*h, ombh2=ombh2, omch2=omch2)
pars.InitPower.set_params(ns=0.96, As=As_init)
pars.set_matter_power(redshifts=[0.], kmax=100.0)

#Linear spectra
pars.NonLinear = model.NonLinear_none
results = camb.get_results(pars)
kh, z, pk = results.get_matter_power_spectrum(
minkh=1e-4, maxkh=10, npoints=npoints)
s8 = np.array(results.get_sigma8())

# considering sigma8 = 0.8228
s8_ratio = sigma/s8[0]
As = s8_ratio**2 * As_init

pars = camb.CAMBparams()
pars.set_cosmology(H0=100*h, ombh2=ombh2, omch2=omch2)
pars.InitPower.set_params(ns=0.96, As=As)
pars.set_matter_power(redshifts=[0.], kmax=2.0)
 #Linear spectra
pars.NonLinear = model.NonLinear_none
results = camb.get_results(pars)
kh, z, pk = results.get_matter_power_spectrum(
minkh=1e-4, maxkh=10,npoints=npoints)
#Non-Linear spectra
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=10, npoints=npoints)

kh8 = 1 / 8 #* h
```

```
# filters
def tophat(x, rf):
    num = np.sin(x*rf) - x*rf*np.cos(x*rf)
    den = (x*rf)**3
    return 3 * num / den

def gaussian(x, rf):
    return np.exp(-(x*rf)**2 * 0.5)

def sharp_k(x, rf):
    _thres = 1 / rf
    _filter = np.ones_like(x)
    _filter[x > _thres] = 0
    return _filter

def sig_squared(filt, x, pk, rf):
    integrand = x**2 * pk *
    filt(x, rf)**2 / np.pi**2 / 2.
    return np.trapz(integrand, x)

def mass_to_r(mass, h, filt='tophat'):
    _rho = om*2.7754e11 * u.Msun / u.Mpc**3 #* h**2

    gam_f = {'tophat': 4*np.pi/3,
            'gaussian': (2*np.pi)**(3/2),
            'sharp_k': 6*np.pi**2}

    r3 = mass / _rho / gam_f[filt]

    return r3**(1/3)

mass = 10**np.linspace(0, 3, npoints) * 1e12*u.Msun
```

```
#The Press-Schechter  function with factor of 2,



rho_mean = om*(3*(100*u.km/u.s/u.Mpc)**2/(8*np.pi*c.G))
.to(u.Msun*u.Mpc**(-3))
delt_c = 3/5*(3/4)**(2/3)*(2*np.pi)**(2/3)

func = lambda x : sig_squared(tophat, kh, pk, x)
r_f = mass_to_r(mass, 1, filt='tophat')
vfunc = np.vectorize(func)
sig2 = vfunc(r_f.value)
sig = np.sqrt(sig2)

dlnsig = np.gradient(np.log(sig))
dlnM = np.gradient(np.log(mass.value))
multi_func_PS = np.sqrt(2/np.pi) * delt_c/sig *
np.exp(-0.5*delt_c**2/sig2)
dNdlnM_PS = multi_func_PS * rho_mean/mass *
np.abs(dlnsig/dlnM)


#Sheth-Tormen  function

A = 0.3222
a = 0.707
P = 0.3
nu = delt_c/sig
multi_func_ST = A*np.sqrt(2*a/np.pi)*(1+(1/a/nu**2)**P)*
nu*np.exp(-a/2*nu**2)

dNdlnM_ST = multi_func_ST * rho_mean/mass *
np.abs(dlnsig/dlnM)
x_axis=np.log10(mass/u.Msun)
```

```
                y_axis=dNdlnM_ST




                #Chi-squared!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                with open("ST_func_x.csv") as f:
                    emp_x = np.loadtxt(f, delimiter=",")[:,1]
                with open("ST_func_y.csv") as f:
                    emp_y = np.loadtxt(f, delimiter=",")[:,1]
                #print(y_axis)
                #print(emp_x)
                indexes=[]
                y_axisu=[]
                for i in emp_x.round(2):
                    k=0
                    for j in x_axis.round(2):
                        if i == j:
                            y_axisu.append(y_axis[k].value)
                            break
                        k=k+1
                #print("\n","index")

                #print(y_axisu)
                #print(x_axis[99])
                chicc=0
                for i in range(25):
                    chicc=(((y_axisu[i]-emp_y[i])**2)/y_axisu[i])+chicc
                chic.append(chicc)
            chi.append(chic)
        print(chi)
        min=10
        indexom=0
        k=0
        indexsigma=0
        for i in chi:
```

```python
    c=0
    for j in chi[k]:
        if chi[k][c]<min:
            min=chi[k][c]
            indexom=k
            indexsigma=c
        c=c+1
    k=k+1
print("\n",indexom)
print("\n",indexsigma)


WD = 'D:/SNU/Cosm'
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(emp_x ,emp_y, c='k', label='from Rockstar')
ax.plot(emp_x, y_axisu, c='b', label='theoretical')
plt.title(f'Graph for the best-fit values')
ax.set_yscale('log')
ax.set_xlabel('$\log M (h^{-1}M_{\odot})$')
ax.set_ylabel('$dN/d\ln M$ [$(h^{-1}$Mpc$)^{-3}$]')
plt.legend()
plt.tight_layout()
plt.savefig(WD+'/FE_N1.png', dpi=300)
# def objective(x, a, b):
#     return a * x + b

# # choose the input and output variables
# x, y = emp_x, emp_y
# # curve fit
# popt, _ = curve_fit(objective, x, y)
# # summarize the parameter values
# a, b = popt
# #print('y = %.5f * x + %.5f' % (a, b))
# # plot input vs output
# plt.plot(x, y)
```

```
            # # define a sequence of inputs between
            # #the smallest and largest known inputs
            # x_line = np.arange(min(x), max(x), 1)
            # # calculate the output for the range
            # y_line = objective(x_line, a, b)
            # # create a line plot for the mapping function

            # y_axisu=[]
            # for i in emp_x.round(2):
            #     k=0
            #     for j in x_axis.round(2):
            #         if i == j:
            #             y_axisu.append(y_axis[k].value)
            #             break
            #         k=k+1
            # plt.plot(x, y_axisu)
            # plt.show()
```

For problem 2:

```
import sys, platform, os
import matplotlib
from matplotlib import pyplot as plt
import numpy as np
import csv
import pandas as pd
from astropy import units as u
from astropy import constants as c


csvf=pd.read_csv('Rockstar_for_CAMB.csv')
mass=csvf["h.Mvir"]
# mas=np.asarray(mass)
# mas10=np.log10(mas)
# mase=np.log(mas)
# print(mase[:3])
N= []
Ma=[]
j=0
step=10**12
M=10**12
while M <(10**15)+1 :
    Ma.append(M)
    N.append(0)
    N[j] = sum(map(lambda x : M+step>x>M, mass))
    j=j+1
    M=M+step
    print(j)
    print('M= ',M)
    if(j==9 or j==118):
        step=step*10
        print('step= ',step)
    if(j==18 or j==119):
        step=step*10
```

```
                    print('step= ',step)


            mas=np.asarray(Ma)
            dN=[]
            dmase=[]
            dN_dmase=[]
            for i in range(len(N)-1):
                dN.append(N[i+1]-N[i])
                dmase.append(np.log(mas)[i+1]-np.log(mas)[i])
                dN_dmase.append(dN[i]/dmase[i])

            dN_dmase.pop(17)
            dN_dmase.pop(8)
            #dmase.pop(19)



            # dN=np.gradient(N)
            # dmase=np.gradient(np.log(mas))

            print('\n',dN)
            print('\n',dmase)
            logm=np.log10(mas)
            # logm=np.delete(logm,-1)
            logm=np.delete(logm,17)
            logm=np.delete(logm,8)
            print('\n',dN_dmase)

            WD = 'D:/SNU/Cosm'
            fig = plt.figure()
            ax = fig.add_subplot(111)
            ax.plot(np.delete(logm,-1) , np.abs(dN_dmase)/(4*(10**8)), c='k')
            plt.title(f'Using MDPL2.Rockstar data')
            ax.set_yscale('log')
            ax.set_xlabel('$\log M (h^{-1}M_{\odot})$')
```

```
            ax.set_ylabel('$dN/d\ln M$ [$(h^{-1}$Mpc$)^{-3}$]')
            plt.tight_layout()
            plt.savefig(WD+'/N3_Rockstar.png', dpi=300)


            #Chi-squared!!!!!!!!!!!!!!!!!!!!!!!!!!
            pd.DataFrame(np.delete(logm,-1)).to_csv("ST_func_x.csv")
            pd.DataFrame(np.abs(dN_dmase)/(4*(10**8))).to_csv("ST_func_y.csv")
```