Homework 3

Qinyue Fei 1901110218@pku.edu.cn

1 Introduction to the Saha Equation

(a)

Considering all electrons are from ionization of neutral Hydrogen, $n_e = n_{H^+}$. Assuming total hydrogen density $n = 10^{20} cm^{-3}$, finding that ionization potential energy of neutral Hydrogen is $\epsilon = 13.6eV$, the partition function of neutral Hydrogen and ionization Hydrogen is $U_H = 2$, $U_{H^+} = 1$. From Saha equation:

$$\begin{aligned}\frac{n_{H^+}}{n_H} &= 2\frac{1}{2}\frac{1}{n_e}\frac{(2\pi m_e kT)^{3/2}}{h^3}e^{-\epsilon/kT} \\ &= \frac{1}{n_{H^+}}\frac{(2\pi m_e kT)^{3/2}}{h^3}e^{-\epsilon/kT}\end{aligned}$$

where m_e is electron mass, k is boltzmann constant, h is planck constant. The H ionization fraction defined as:

$$x = \frac{n_{H^+}}{n_H + n_{H^+}}$$

Finding that gas pressure is $P_{gas} = (n_H + n_{H^+} + n_e)kT$, and gas density is $\rho = (n_H + n_{H^+})m_u$, so the Saha equation can be written as:

$$\frac{x^2}{1-x} = \frac{1}{n}\frac{(2\pi m_e kT)^{3/2}}{h^3}e^{-\epsilon/kT}$$

So the H ionization fraction versus temperature is shown as figure 1.

(b)

Searching the partition function and ionization energy of all ionization states of C:

	C	C^+	C^{2+}	C^{3+}	C^{4+}	C^{5+}	C^{6+}
Partition Function	6	6	1	2	1	2	1
Ionization Energy [eV]	11.26030	24.38332	47.8878	64.4939	392.087	489.99334	nan

where 'nan' means there is no electrons over carbon nucleon so the ionization energy is infinity. And the neutral and the fraction of all ionization states of C versus temperature is shown as figure 1.

(c)

My x-axis is in log because the temperature is across a large range of magnitude so x-axis in log will show better. My y-axis is in linear because the fraction is between 0 and 1. The linear axis is better.

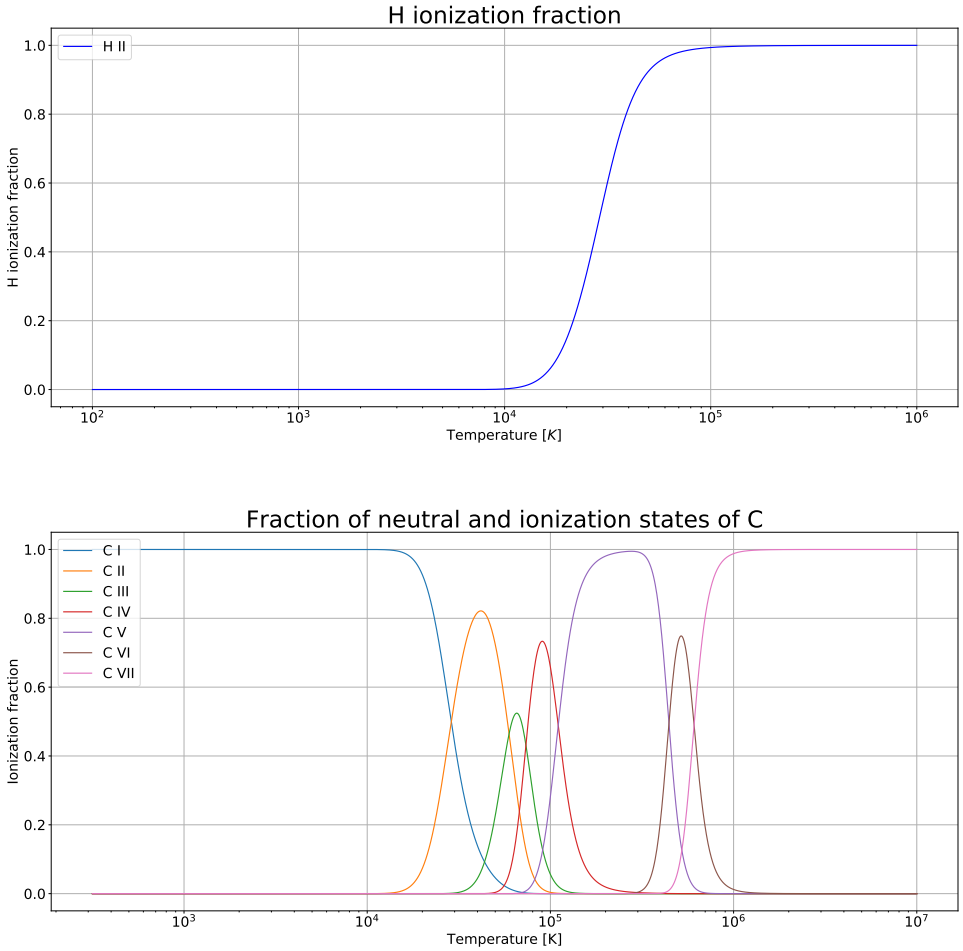

Figure 1: Figure above shows the H ionization fraction. The under figure shows fraction of neutral and ionization states of C.

(d)

The neutral carbon is always appears in temperature of a few thousand kelvin. The temperature is about $3 \times 10^4 K$. When the C VI is main distribution of carbon, the temperature is about $3 \times 10^5 K$ so the temperature of coronae is about $3 \times 10^5 K$.

2 Equation of State

(a)

i:

The momentum distribution of non-relativistic ideal gas $n(p)$ is given by $Maxwell - Boltzmann$ distribution:

$$n(p)dp = \frac{n}{(2\pi mkT)^{3/2}} e^{-p^2/2mkT} 4\pi p^2 dp$$

where m is particle mass, p is momentum, T is temperature of gas, k is Boltzmann constant.

Consider relation between pressure P and momentum:

$$\begin{aligned}
P &= \frac{1}{3}\int_0^\infty pv_p n(p)dp \\
&= \frac{1}{3}\int_0^\infty \frac{p^2}{m}\frac{n}{(2\pi mkT)^{3/2}}e^{-\frac{p^2}{2mkT}}4\pi p^2 dp \\
&= \frac{1}{3m}\frac{2\pi n(2mkT)^{5/2}}{(2mkT)^{3/2}\pi^{3/2}}\int_0^\infty \left(\frac{p^2}{2mkT}\right)^{3/2} e^{-\frac{p^2}{2mkT}} d\frac{p^2}{2mkT} \\
&= \frac{4}{3\sqrt{\pi}}nkT\cdot\Gamma\left(\frac{5}{2}\right) \\
&= nkT \\
&= \frac{R}{\mu}\rho T
\end{aligned}$$

where v_p is particle velocity and in non-relativistic gas $v_p = p/m$, $\Gamma(x)$ is Gamma function, μ is relative atomic mass.
For degenerate electrons which are satisfied Fermi distribution:

$$\begin{aligned}
n_e(p) &= \frac{8\pi p^2}{h^3}, \quad p \le p_F \\
n_e(p) &= 0, \quad p > p_F
\end{aligned}$$

where p_F is Fermi momentum which is defined by $n_e = \int_0^{p_F} n_e(p)dp$ and $p_F = \left(\frac{3n_e}{8\pi}\right)^{1/3} h$ where $n_e = \frac{\rho}{m_u}$ is electron number density. If the electrons are non-relativistic $v \ll c$, $v = p/m$, the EoS of degenerate electrons is:

$$\begin{aligned}
P_{e,NR} &= \frac{1}{3}\int_0^{p_F} \frac{8\pi p^4}{m_e h^3}dp \\
&= \frac{8\pi}{15m_e h^3}p_F^5 \\
&= \frac{h^2}{20m_e}\left(\frac{3}{\pi}\right)^{\frac{2}{3}}\left(\frac{\rho}{m_u}\right)^{\frac{5}{3}} \\
&= K_{NR}\left(\frac{\rho}{\mu_e}\right)^{5/3}
\end{aligned}$$

where m_e is mass of electron particle, ρ is density of electrons, $K_{NR} = \frac{h^2}{20m_e m_u^{5/3}}\left(\frac{3}{\pi}\right)^{2/3} = 1.0036 \times 10^{13}[cgs]$. If the electrons are relativistic particles, $v = c$:

$$\begin{aligned}
P_{e,ER} &= \frac{1}{3}\int_0^{p_F} \frac{8\pi c p^3}{h^3}dp \\
&= K_{ER}\left(\frac{\rho}{\mu_e}\right)^{\frac{4}{3}}
\end{aligned}$$

where $K_{ER} = \frac{hc}{8m_u^{\frac{4}{3}}}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} = 1.2435 \times 10^{15}[cgs]$.

The boundary between ideal gas and degenerate electrons is defined by $P_{e,NR} = P$ and $P_{e,ER} = P$:

$$logT = \frac{2}{3}log\rho + log\frac{K_{NR}\mu}{R\mu_e^{5/3}}, \quad (non - relativistic)$$
$$logT = \frac{1}{3}log\rho + log\frac{K_{ER}\mu}{R\mu_e^{4/3}}, \quad (relativistic)$$

ii:
For radiation: $p = \epsilon/c$ where $\epsilon = h\nu$ is energy of each photon and the momentum distribution is satisfied to Planck function so the pressure is:

$$\begin{aligned} P_{rad} &= \frac{1}{3}\int_0^\infty pcn(p)dp \\ &= \frac{1}{3}aT^4 \end{aligned}$$

where $a = \frac{8\pi^5 k^4}{15h^3c^3} = 7.56 \times 10^{-15}$ erg cm^{-3}K^{-4} is a constant. So the boundary between radiation and ideal gas is defined by $P_{rad} = P$:

$$logT = \frac{1}{3}log\rho + \frac{1}{3}log\frac{3R}{a\mu}$$

iii:
By $P_{e,NR} = P_{e,ER}$, the transition takes place at densities around:

$$\rho \approx \mu_e m_u \frac{125\pi}{24}\left(\frac{m_e c}{h}\right)^3 \approx 10^6 \mu_e g cm^{-3}$$

iv:
The Coulomb parameter Γ_C is defined as:

$$\Gamma_C = \frac{Z^2e^2}{kT}\left(\frac{4\pi\rho}{3Am_u}\right)^{\frac{1}{3}}$$

For crystallization, $\Gamma_C \approx 170$, so the transition is:

$$logT = \frac{1}{3}log\rho + log\frac{Z^2e^2}{k\Gamma_C}\left(\frac{4\pi}{3Am_u}\right)^{\frac{1}{3}}$$

where Z is proton number, A is nuclear number. For Hydrogen, $Z = 1$, $A = 1$, $\mu_e = 1$, $m_u = 1.67 \times 10^{-24}$g.

(b)

The density and temperature of center of a low mass white dwarf is $\rho_c = 10^4 g \cdot cm^{-3}$, $T_c = 10^6 K$.
The density and temperature of center of a high mass white dwarf is $\rho_c = 10^8 g \cdot cm^{-3}$, $T_c = 10^8 K$.
The density and temperature of center of sun is $\rho_c = 1.622 \times 10^2 g \cdot cm^{-3}$, $T_c = 1.571 \times 10^7 K$.

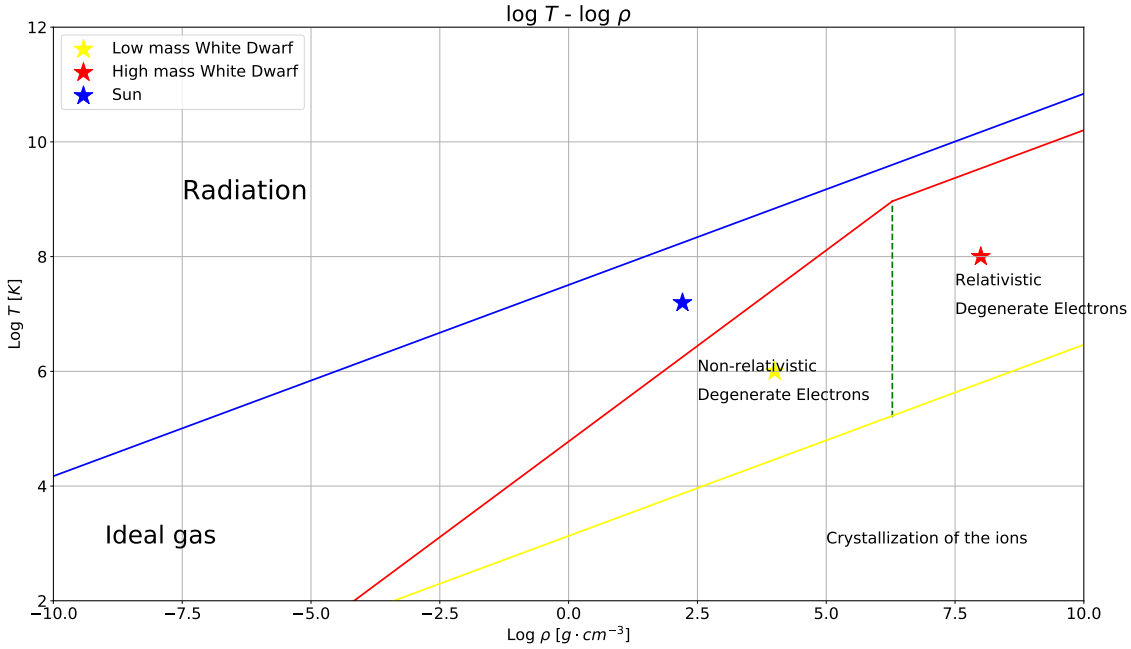

Figure 2: Contours in the log$T$ - logρ planes. Locations of low mass white dwarf, high mass white dwarf and sun.

3 The Chandrasekhar Mass

(a) The Equation of State of completely degenerate electrons in the extreme relativistic limit is:

$$P_{ER} = \frac{K_{ER}}{\mu_e^{4/3}} \rho^{\frac{4}{3}}$$

The relationship between mass and radius for polytropes is:

$$K = N_n G M^{(n-1)/n} R^{(3-n)/n}$$

$$N_n = \frac{(4\pi)^{1/n}}{n+1} \Theta_n^{(1-n)/n} z_n^{(n-3)/n}$$

where n is defined by $n = \frac{1}{\gamma - 1}$ and in this case $n = 3$, K is constant and $K = \frac{K_{ER}}{\mu_e^{4/3}}$. Thus the mass is equals to :

$$\begin{aligned} M &= \left(\frac{K_{ER}}{\mu_e^{4/3} G N_3} \right)^{3/2} \\ &= \frac{4 K_{ER}}{\mu_e^{4/3} G (4\pi)^{1/3} \Theta_3^{-2/3}}^{3/2} \\ &= \frac{4}{\sqrt{\pi} \mu_e^2} \left(\frac{K_{ER}}{G} \right)^{3/2} \Theta_3 \end{aligned}$$

From textbook we find that $\Theta_3 = 2.01824$, thus the mass is:

$$M_{ch} = \frac{5.829}{\mu_e^2} M_\odot \tag{1}$$

(b)
For only He, $Z = 2$, $A = 4$, $X = 1$, where Z is atomic number, A is nucleon number, X is abundance. thus $\mu_e = \left(\frac{ZX}{A}\right)^{-1} = 2$, thus:

$$M_{ch} = \frac{5.829}{mu_e^2} M_\odot \approx 1.457 M_\odot \tag{2}$$

(c) For only C and O:

$$Z_C = 6, \quad Z_O = 8, \quad A_C = 12, \quad A_O = 16, \quad X_C = 80\%, \quad X_O = 20\%.$$

Thus:

$$\mu_e = \left(\frac{Z_C X_C}{A_C} + \frac{Z_O X_O}{A_O}\right)^{-1} = 2$$

So the Chandrasekhar mass limit is:

$$M_{ch} = \frac{5.829}{\mu_e} M_\odot \approx 1.457 M_\odot$$

4 Energy Generation in the Sun

(a)
The energy produced during each reaction is:

$$\epsilon = 4 \times m_H c^2 - m_{He} c^2 - 2 \times m_e c^2 - 2 \times m_{\nu_e} c^2$$

Considering the annihilation between e^+ and e^-, where m_{ν_e} is so small that can be ignored. So the energy is:

$$\begin{aligned}
\epsilon &= 4 \times m_H c^2 - m_{He} c^2 + 2 \times m_e c^2 - 2 \times m_{\nu_e} c^2 \\
&= (4 \times 1.00784 - 4.002602) \times 1.674 \times 10^{-27} \times (3 \times 10^8)^2 + 2 \times 9.10956 \times (3 \times 10^8)^2 \\
&\approx 4.497 \times 10^{-12} J
\end{aligned}$$

The luminosity of sun is $L_\odot = 3.828 \times 10^{26}$W. So the reaction times per second is:

$$N = \frac{L_\odot}{\epsilon} \approx 8.513 \times 10^{37}$$

(b) Each pp-chain produces one ^{4}He particle, so the velocity of mass production is:

$$M_{He} = N \times m_{He} \approx 5.704 \times 10^{11}\ kg$$

(c) Each reaection produces two photons so the energy of each photon is:

$$\epsilon_\gamma = \frac{\epsilon}{2} \approx 13.01\ MeV$$

(d) Each reaction produces two neutrinos so the neutrino flux is:

$$f = \frac{2N}{4\pi r^2}$$
$$= \frac{2 \times 9.183 \times 10^{37}}{4\pi(1.5 \times 10^{13})^2} \approx 6.495 \times 10^{10}\ cm^{-2}s^{-1}$$

5 Opacities

(a)

At low density and high temperature, where ρ ranges in $10^{-10} - 10^{-5}\ g \cdot cm^{-3}$ and T ranges in $10^6 - 10^8$K, opacity has a constant value because the main distribution is electron scattering.

At low density and lower temperature, where ρ ranges in $10^{-10} - 10^{-3}\ g \cdot cm^{-3}$ and T ranges in $10^4 - 10^6$K, due to free-free and bound-free absorptions, opcaity is proportional to $\rho T^{-7/2}$.

At temperature $3 \times 10^3 < T < 6 \times 10^3$K, density ρ ranges in $10^{-10} - 10^{-5}\ g \cdot cm^{-3}$, metallicity Z ranges in $0.001 - 0.02$, negative hydrogen ion H^- is main distribution to opacity.

At lower temperature $T < 4000$K, and higher density where $10^{-5} < \rho < 10^0\ g \cdot cm^{-3}$, molecules and dust is the main reason for opacity.

At low density where $\rho < 10^0\ g \cdot cm^{-3}$ and high temperature $T > 10^6$K, the opacity is dominated by the conductivity of degenerate electrons and decreases strongly with increasing ρ.

(b)

Comparing the opacity curve for $log\rho$ = -6 in the left panel of Fig. 5.2 and the opacity dominated by four processes. We can find that temperature T ranges in $10^3 - 10^4$K, the H ion fits the realistic opacity curve best. At temperature T ranges in $10^4 - 10^5$K, the free-free absorption approximation fits curve well. At T ranges in $10^5 - 2 \times 10^5$K, the bound- free absorption fits the curve. At $T > 2 \times 10^5$K, the electron scattering fits the realistic curve.

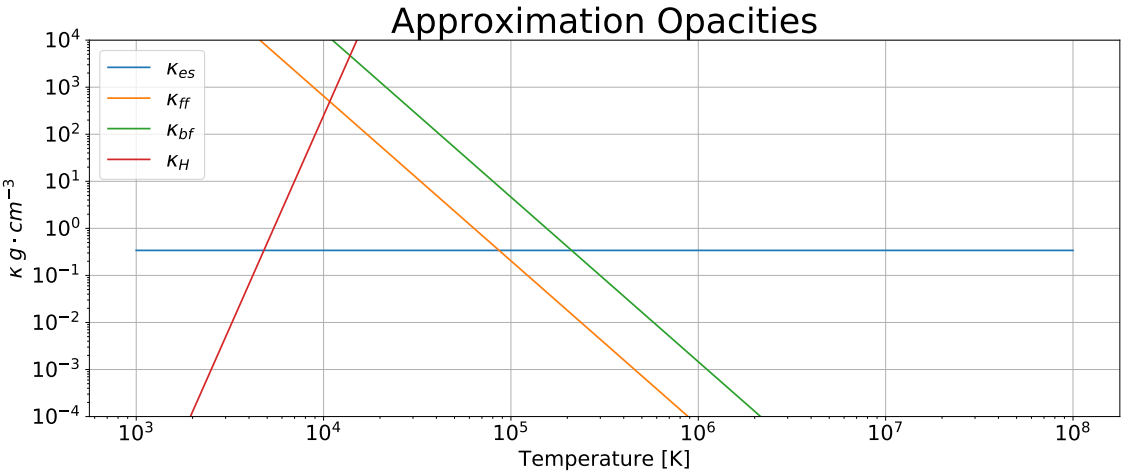

Figure 3: The opacity of different processes.

(c)

For one solar mass sequence:

At $T = 10^7$K, the density $log\ \rho \approx 2$, $log\ \kappa \approx 0$. Thus the mean free path is $l_{mfp} = \frac{1}{\rho\kappa} \approx 0.01cm$.
At $T = 10^5$K, $log\ \rho \approx -2.5$, $log\ \kappa \approx 4$, so the mean free path $l_{mfp} = \frac{1}{\rho\kappa} \approx 0.03$cm.
At $T = 10^4$K, $log\ \rho \approx -6.3$, $log\ \kappa \approx 1$, mean free path $l_{mfp} = \frac{1}{\rho\kappa} \approx 2 \times 10^5$cm.
(d) For the definition of Rosseland mean opacity κ, suppose the frequency-dependent opacity coefficient $\kappa_\nu = \kappa_0 \nu^{-\alpha}$:

$$\begin{aligned}
\frac{1}{\kappa} &= \frac{\int_0^\infty \frac{1}{\kappa_\nu}\frac{\partial U_\nu}{\partial T}d\nu}{\int_0^\infty \frac{\partial U_\nu}{\partial T}d\nu} \\
&= \frac{\frac{d}{dT}\left(\int_0^\infty \frac{\nu^\alpha}{\kappa_0}U_\nu d\nu\right)}{\frac{d}{dT}\left(\int_0^\infty U_\nu d\nu\right)} \\
&= \frac{\frac{d}{dT}\left(\frac{kT}{h}\right)^{4+\alpha}\int_0^\infty \frac{x^{4+\alpha}}{e^x-1}dx}{\frac{d}{dT}\left(\frac{kT}{h}\right)^4\int_0^\infty \frac{x^3}{e^x-1}dx} \\
&= C \cdot T^\alpha
\end{aligned}$$

where C is a constant. So the Rosseland mean opacity depends on the temperature $\kappa \propto T^{-\alpha}$

6 The Initial Mass Function

I chose those stars whose errors are not so much, made the HR diagram of these stars, found many stars are on the white dwarf branch and giant branch, then I deleted these data and got figure 4. Then I got simulated stars data from CMD wedsite and fits the relationship between star mass and color, absolute magnitude:

$$log10(M) = 0.1986(B_P - R_P) - 0.1761G + 0.66656$$

I used this relationship calculate mass of each star in gaia data. I divide these stars into 30 groups as their mass. Noticing that number of stars whose mass less than $0.1M_\odot$ is so small that doesn't have statistical meaning. I deleted them. Considering that stars with small mass are difficult to be observed so only within small radius, stars with little mass can be observed completely. So I chose the 'complete radius' within which stars can be observed completely for stars whose mass is small. Considering lifetime of stars with large mass is shorter than small mass stars, some massive stars have dead. Considering local stars are all on the spiral arms, the formation time of stars is shorter than hubble time. I assumed this time is 50 Gyr. Then I assumed that star formation rate is constant during 50 Gyr and calculate total stars formed during 50 Gyr. After these correction, I got figure 5. There are many differences between two results. This IMF is applied to star mass less than $2M_\odot$ while Hipparcos IMF is applied to mass higher than $10M_\odot$. The number densities of stars whose mass range in $0.3M_\odot - 3M_\odot$ is very different. I think the reason is that Hipparcos can not observe so much stars while gaia can observe them.

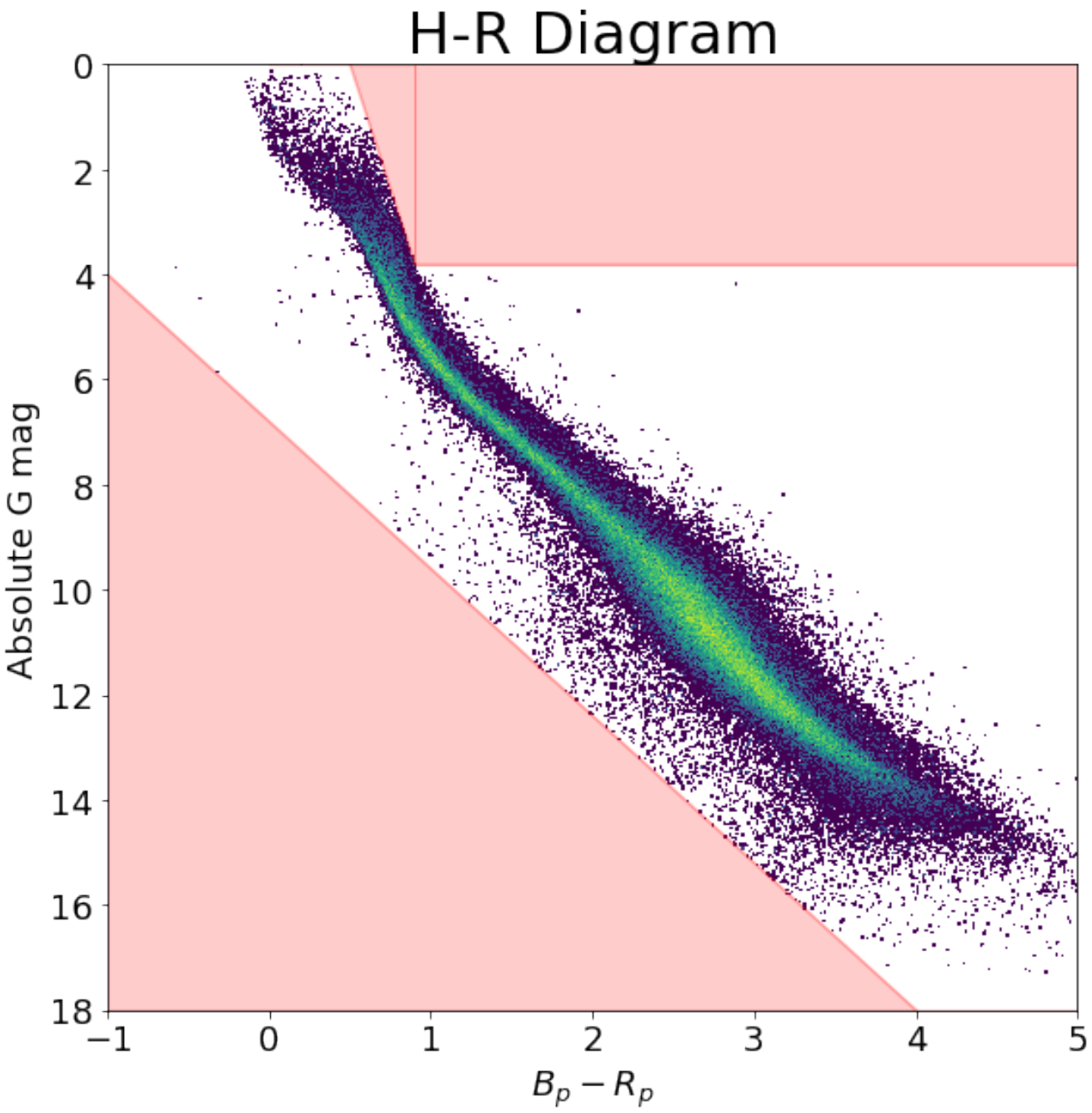

Figure 4: Gaia data without white dwarfs and giant stars.

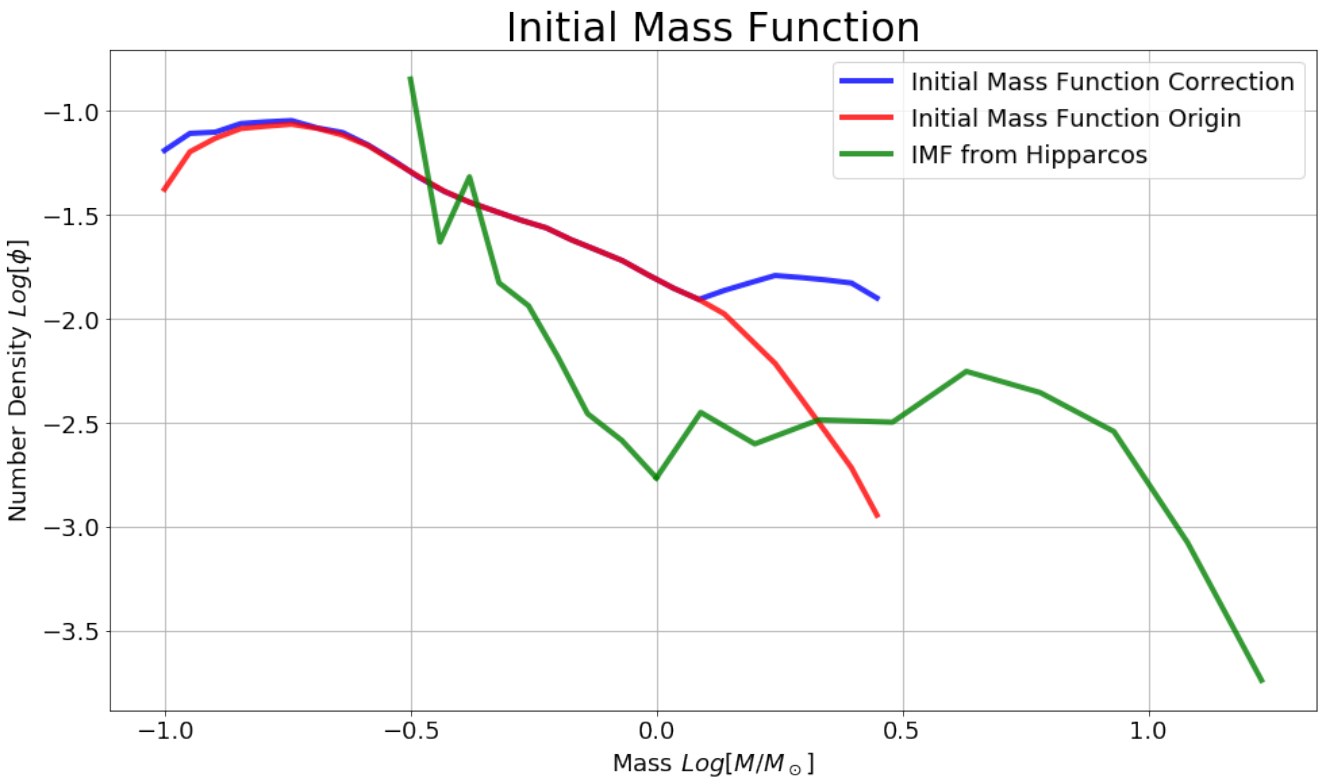

Figure 5: The initial mass function calculated from gaia data. Green line is IMF curve from Hipparcos diagram.

7 Code

7.1 Introduction to the Saha Equation

```
import numpy as np
from scipy import integrate
import matplotlib.pyplot as plt
e = 1.602e-19 #
k=8.617e-5 #cgs
ne = 1e20
me=511e3 #cgs
h = 4.136*10**(-15) #cgs
c = 3e10 #cgs
g = {}
phi = {}
g['H'] = [0,2,1]
phi['H'] = [0,13.59844]
g['C'] = [0,6,6,1,2,1,2,1]
#phi['C'] = [13.61806,35.11730,54.9355,77.41353,113.8990,138.1197,739.29]
phi['C'] = [11.26030,24.38332,47.8878,64.4939,392.087,489.99334]
def f(Z, l):
    g = {}
    ne = 1e20
    g['H'] = [0,2,1]
    phi['H'] = [0,13.59844]
    g['C'] = [0,6,6,1,2,1,2,1]
    #phi['C'] = [0,13.61806,35.11730,54.9355,77.41353,113.8990,138.1197]
    phi['C'] = [0,11.26030,24.38332,47.8878,64.4939,392.087,489.99334]
    #A =  2*g[Z][l+1]/g[Z][l]*np.exp(-13.6/k/T*((l+1)**(-2)-l**(-2))-phi[Z][l]/k/T)/ne
    A = 2*g[Z][l+1]/g[Z][l]*np.exp(-phi[Z][l]/k/T)/ne
    B = (2*np.pi*me*k*T/c**2)**1.5/h**3
    return A*B

T = np.logspace(2,5,1000)
F = f('H',1)
x = (-F+(F**2+4*F)**0.5)/2
plt.figure(figsize=(21,9))
plt.plot(T,x,'b',label='H II')
plt.semilogx()
plt.xlabel('Temperature [$K$]',fontsize=18)
plt.ylabel('H ionization fraction',fontsize=18)
plt.title('H ionization fraction',fontsize=30)
plt.grid()
plt.legend(loc='upper left',fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
```

```
T = np.logspace(2.5,6.5,1000)
L = [0,1,2,3,4,5,6]
f1 = f('C',1)
f2 = f('C',2)
f3 = f('C',3)
f4 = f('C',4)
f5 = f('C',5)
f6 = f('C',6)
F1 = f1
F2 = f1*f2
F3 = f1*f2*f3
F4 = f1*f2*f3*f4
F5 = f1*f2*f3*f4*f5
F6 = f1*f2*f3*f4*f5*f6
Nc1 = ne/(F1+2*F2+3*F3+4*F4+5*F5+6*F6)
Nc2 = Nc1*F1
Nc3 = Nc1*F2
Nc4 = Nc1*F3
Nc5 = Nc1*F4
Nc6 = Nc1*F5
Nc7 = Nc1*F6
ff1 = Nc1/(Nc1+Nc2+Nc3+Nc4+Nc5+Nc6+Nc7)
ff2 = Nc2/(Nc1+Nc2+Nc3+Nc4+Nc5+Nc6+Nc7)
ff3 = Nc3/(Nc1+Nc2+Nc3+Nc4+Nc5+Nc6+Nc7)
ff4 = Nc4/(Nc1+Nc2+Nc3+Nc4+Nc5+Nc6+Nc7)
ff5 = Nc5/(Nc1+Nc2+Nc3+Nc4+Nc5+Nc6+Nc7)
ff6 = Nc6/(Nc1+Nc2+Nc3+Nc4+Nc5+Nc6+Nc7)
ff7 = Nc7/(Nc1+Nc2+Nc3+Nc4+Nc5+Nc6+Nc7)
plt.figure(figsize=(21,9))
plt.plot(T,ff1,label='C I')
plt.plot(T,ff2,label='C II')
plt.plot(T,ff3,label='C III')
plt.plot(T,ff4,label='C IV')
plt.plot(T,ff5,label='C V')
plt.plot(T,ff6,label='C VI')
plt.plot(T,ff7,label='C VII')
plt.legend(loc='upper left',fontsize=18)
plt.xlabel('Temperature [K]',fontsize=18)
plt.ylabel('Ionization fraction',fontsize=18)
plt.title('Fraction of neutral and ionization states of C',fontsize=30)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.semilogx()
plt.grid()
```

7.2 Equation of State

```
me = 9.11*10**(-28)
mu = 1.67*10**(-24)
#mu_e = me/0.5/mu
mu_e = 1
h = 1.38*10**(-16)
c = 3*10**10
e = 4.83*10**(-10)
Z = 1
A = 1
E = 170
h = 6.626*10**(-27)
u = 0.5
mu = 1.67*10**(-24)
me = 9.11*10**(-28)
k = 1.38*10**(-16)
R = k/mu
print(R/10**7)
K_NR = h**2/20/me/mu**(5/3)*(3/np.pi)**(2/3)
print(K_NR/10**12)
K_ER = h*c/8/mu**(4/3)*(3/np.pi)**(1/3)
print(K_ER/10**15)
#K_NR = 1.0036*10**13
#K_ER = 1.2435*10**15
#K_NR = h**2/20/me/mu**(5/3)*(3/np.pi)**(2/3)
#K_ER = h*c/8/mu**(4/3)*(3/np.pi)**(1/3)
R = 8.314*10**7
a = 7.56*10**(-15)
rho_g = np.log10(mu_e*(K_ER/K_NR)**3)
C = 1*(4.83*10**(-10))**2/k/170*(4*np.pi/3/mu)**(1/3)
plt.figure(figsize=(21,12))
rho = np.linspace(-10,10,1000)
rho1 = np.linspace(-10,rho_g,1000)
rho2 = np.linspace(rho_g,10,1000)
T1 = 2/3*rho1+np.log10(K_NR*0.5/R/mu_e**(5/3))
T2 = 1/3*rho2+np.log10(K_ER*0.5/R/mu_e**(4/3))
T3 = 1/3*rho+1/3*np.log10(3*R/a)
T4 = 1/3*rho+np.log10(C)
plt.plot(rho1,T1,'r',linewidth=2)
plt.plot(rho2,T2,'r',linewidth=2)
plt.plot(rho,T3,'b',linewidth=2)
plt.plot(rho,T4,'yellow',linewidth=2)
plt.vlines(rho_g,0,8.9,'green','--',linewidth=2)
plt.ylim(2,12)
plt.xlim(-10,10)
plt.xlabel(r'Log $\rho$ [$g \cdot cm^{-3}$]',fontsize=18)
```

```
plt.ylabel(r'Log $T$ [$K$]',fontsize=18)
plt.title(r'log $T$ - log $\rho$',fontsize=24)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.text(-7.5,9,'Radiation',fontsize=30)
plt.text(-9,3,'Ideal gas',fontsize=28)
plt.text(2.5,6,'Non-relativistic',fontsize=18)
plt.text(2.5,5.5,'Degenerate Electrons',fontsize=18)
plt.text(7.5,7.5,'Relativistic',fontsize=18)
plt.text(7.5,7,'Degenerate Electrons',fontsize=18)
plt.text(5,3,'Crystallization of the ions',fontsize=18)
x1,y1 = np.array(4),np.array(6)
x2,y2 = 8,8
x3,y3 = np.log10(1.622*10**2),np.log10(1.571*10**7)
plt.scatter(x1,y1,s=np.array(500),c='yellow',marker='*',label='Low mass White Dwarf')
plt.scatter(x2,y2,s=np.array(500),c='red',marker='*',label='High mass White Dwarf')
plt.scatter(x3,y3,s=np.array(500),c='blue',marker='*',label='Sun')

plt.legend(loc='upper left',fontsize=18)
plt.grid()
```

7.3 The Chandrasekhar Mass

```
me = 9.11*10**(-28)
mu = 1.67*10**(-24)
#mu_e = me/0.5/mu
mu_e = 1
h = 1.38*10**(-16)
c = 3*10**10
e = 4.83*10**(-10)
Z = 1
A = 1
E = 170
h = 6.626*10**(-27)
u = 0.5
mu = 1.67*10**(-24)
me = 9.11*10**(-28)
k = 1.38*10**(-16)
G = 6.67*10**(-8)
R = k/mu
#print(R/10**7)
K_NR = h**2/20/me/mu**(5/3)*(3/np.pi)**(2/3)
#print(K_NR/10**12)
K_ER = h*c/8/mu**(4/3)*(3/np.pi)**(1/3)
#K_ER = 1.2435*10**15
Mch = 4*(K_ER/G)**(3/2)/np.sqrt(np.pi)*2.01824/1.989/10**33
```

7.4 Energy generation in the sun

```
m_H = 1.00784
m_He = 4.002602
m_p = 1.674*10**(-27)
c_e = 1.602*10**(-19)
c = 3*10**8
m_e = 9.10956*10**(-31)
L_sun = 3.828*10**26
epsilon = (4*m_H-m_He)*m_p*c**2-2*m_e*c**2
r = 1.5*10**13
L_sun/epsilon
L_sun/epsilon*m_He*m_p/1e11
epsilon/2/c_e/1e6
2*L_sun/epsilon/4/np.pi/r**2/1e10
```

7.5 Opacities

```
X = 0.7
Z = 0.02
rho = 10**(-6)
T = np.logspace(3,8,1000)
kappa_es = 0.2*(1+X)*T**0
kappa_ff = 3.8*10**22*(1+X)*rho*T**(-7/2)
kappa_bf = 4.3*10**25*(1+X)*Z*rho*T**(-7/2)
kappa_H = 2.5*10**(-31)*(Z/0.02)*rho**(1/2)*T**9

plt.figure(figsize=(16,6))
plt.plot(T,kappa_es,label = r'$\kappa_{es}$')
plt.plot(T,kappa_ff,label = r'$\kappa_{ff}$')
plt.plot(T,kappa_bf,label = r'$\kappa_{bf}$')
plt.plot(T,kappa_H,label = r'$\kappa_{H}$')
plt.ylim(1e-4,1e4)
plt.loglog()
plt.grid()
plt.xlabel('Temperature [K]',fontsize=18)
plt.ylabel(r'$\kappa\; g\cdot cm^{-3}$',fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.legend(loc='upper left',fontsize=18)
plt.title('Opacities',fontsize=30)
```

7.6 The Initial Mass Function

```
from astropy.io import fits
gaia = fits.open('gaia_all100pc-result.fits')
data = pd.DataFrame(gaia[1].data)
data['distance'] = 1000/data['parallax']
data['g_abs'] = data['phot_g_mean_mag']+5-5*np.log10(data['distance'])
data = data[data['astrometric_excess_noise']<1]
D = np.arange(1,100,1)
G = np.linspace(-1,27,100)
N = []  #number density
for i in range(len(G)-1):
    dat = data[(data['g_abs']>G[i])&(data['g_abs']<G[i+1])]
    dmax = dat['distance'].max()
    n = len(dat)
    N.append(3*n/4/np.pi/dmax**3)
data = data[data['phot_bp_mean_flux_error']>0]
data = data[data['phot_g_mean_flux_error']>0]
data = data[data['phot_rp_mean_flux_error']>0]
data = data[data['phot_bp_mean_flux']>data['phot_bp_mean_flux_error']]
data = data[data['phot_rp_mean_flux']>data['phot_rp_mean_flux_error']]
data = data[data['phot_g_mean_flux']>data['phot_g_mean_flux_error']]
import matplotlib.colors as colors
color = np.linspace(-1,5,1000)
m_abs = 14/5*color+34/5
y1 = 18+0*color
m = 28/11*color+14/11
x = np.linspace(0,0.9,1000)
y = 19/2*x-19/4
y2 = 0+0*x
x1 = np.linspace(0.9,5)
y3 = 0+0*x1
y4 = 3.8+0*x1
plt.figure(figsize=(9,9))
plt.hist2d(data['bp_rp'],data['g_abs'],bins=1000,norm = colors.LogNorm())
plt.plot(color,m_abs,'r',alpha=0.2)
#plt.plot(color,m,'b')
plt.plot(x,y,'r',alpha=0.2)
plt.ylim(0,18)
plt.xlim(-1,5)
plt.hlines(3.8,0.9,5,'r',alpha=0.2)
#plt.vlines(0.9,0,3.7,'r')
plt.xlabel(r'$B_p-R_p$',fontsize=18)
plt.ylabel('Absolute G mag',fontsize=18)
plt.title('H-R Diagram',fontsize=30)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
```

```
plt.gca().invert_yaxis()
plt.fill_between(color,m_abs,y1,color='r',alpha=0.2)
plt.fill_between(x,y,y2,color='r',alpha=0.2)
plt.fill_between(x1,y3,y4,color='r',alpha=0.2)
plt.show()
data_c = data[data['g_abs']<14/5*data['bp_rp']+34/5]
data_1 = data_c[data_c['g_abs']>19/2*data_c['bp_rp']-19/4]
data_2 = data_c[(data_c['g_abs']>4)&(data_c['g_abs']<19/2*data_c['bp_rp']-19/4)]
frames=[data_1,data_2]
result=pd.concat(frames)
color = np.linspace(-1,5,1000)
m_abs = 14/5*color+34/5
x = np.linspace(0,0.9)
y = 19/2*x-19/4
color = np.linspace(-1,5,1000)
m_abs = 14/5*color+34/5
y1 = 18+0*color
m = 28/11*color+14/11
x = np.linspace(0,0.9,1000)
y = 19/2*x-19/4
y2 = 0+0*x
x1 = np.linspace(0.9,5)
y3 = 0+0*x1
y4 = 3.8+0*x1
plt.figure(figsize=(9,9))
#plt.hist2d(data_2['bp_rp'],data_2['g_abs'],bins=1000)#,norm = colors.LogNorm())
#plt.hist2d(data_1['bp_rp'],data_1['g_abs'],bins=1000)#,norm = colors.LogNorm())
plt.hist2d(result['bp_rp'],result['g_abs'],bins=1000,norm=colors.LogNorm())
plt.plot(color,m_abs,'r',alpha=0.2)
plt.plot(x,y,'r',alpha=0.2)
plt.ylim(0,18)
plt.xlim(-1,5)
plt.hlines(3.8,0.9,5,'r',alpha=0.2)
#plt.vlines(0.9,0,3.8)
plt.fill_between(color,m_abs,y1,color='r',alpha=0.2)
plt.fill_between(x,y,y2,color='r',alpha=0.2)
plt.fill_between(x1,y3,y4,color='r',alpha=0.2)
plt.xlabel(r'$B_p-R_p$',fontsize=18)
plt.ylabel('Absolute G mag',fontsize=18)
plt.title('H-R Diagram',fontsize=30)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.gca().invert_yaxis()
data = np.loadtxt('1.dat')
data_c = data[np.where(data[:,-3]>4)]
G = data_c[:,-3]
bp = data_c[:,-2]
rp = data_c[:,-1]
```

```
bp_rp = bp[:] - rp[:]
plt.figure(figsize=(9,9))
plt.hist2d(result['bp_rp'],result['g_abs'],bins=1000,norm=colors.LogNorm())
plt.hist2d(bp_rp,G,bins=1000,norm = colors.LogNorm())
#plt.hist2d(data['bp_rp'],data['g_abs'],bins=100,norm = colors.LogNorm())
plt.gca().invert_yaxis()
plt.xlabel(r'$B_p-R_p$',fontsize=18)
plt.ylabel('Absolute G mag',fontsize=18)
plt.title('H-R Diagram',fontsize=30)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
import scipy.optimize as opt
bp_rp = data_c[:,-2]-data_c[:,-1]+0.5
G = data_c[:,-3]
mass = np.log10(data_c[:,3])
def f1(x,a,b,c):
    return a*x[0]+b*x[1]+c
def f(x,a,b,c,d,e,f):
    return a*x[0]+b*x[1]+c+d*x[0]**2+e*x[1]**2+f*x[0]*x[1]
k1 = opt.curve_fit(f1,[bp_rp,G],mass)[0]
k = opt.curve_fit(f,[bp_rp,G],mass)[0]
print(k1,k)
result['mass'] =
    k[0]*result['bp_rp']+k[1]*result['g_abs']+k[3]*result['bp_rp']**2+k[4]*result['g_abs']**2+k[
result['Mass'] = k1[0]*result['bp_rp']+k1[1]*result['g_abs']+k1[2]
catalog = fits.open('I_239_hip_main.dat.gz.fits')
data = catalog[1].data #read the data
s = np.array([data['Vmag'], data['B-V'], data['Plx'], data['e_Plx'], data['HIP']])
    #get the data about apparent magnitude, color(B-V) and angle
ele = np.where(s[2]>3 * s[3])[0]
dat = s.T[ele].T
Vmag = dat[0]
color = dat[1]
Plx = dat[2]
e_Plx = dat[3]
d = np.tan(np.pi/(180*3600))/np.tan(Plx * np.pi/(180*3600*1000))
D = 10
M_abs = Vmag - 5 * np.log10(d/D) #calculate the absolute magnitude of these stars
bot_lim = -4
top_lim = 15
step = 1
bins = np.arange(bot_lim,top_lim,step)
N = []
cou = []
for i in range(len(bins)-1):
    n = len(np.where((M_abs>bins[i]) & (M_abs<bins[i+1]))[0]) #The number of stars
        in each bin
```

```
    #r = np.sum(d[np.where((M_abs>bins[i]) & (M_abs<bins[i+1]))[0]])/n #The average
        radius of every star in each bin
    r = d[np.where((M_abs>bins[i]) & (M_abs<bins[i+1]))[0]].max()
    N.append(3*n/(4*np.pi*r**3))
    cou.append(len(np.where((M_abs>bins[i]) & (M_abs<bins[i+1]))[0])) #Count the
        number of stars

bins_c = np.arange(bot_lim+step/2,top_lim-step/2,step)
N = np.array(N)
R = np.array([0.1,0.25,0.48,0.51,0.43,0.40,0.60,0.70,0.90,1,1,1,1,1,1,1,1,1]) #The
    correction of luminosity function.
N_r = N * R

N_co = N * 10**(-0.3*(bins_c-4.83))

mass_bin = np.array([2.23, 2.08, 1.93, 1.78, 1.63, 1.48, 1.33, 1.20, 1.09, 1.00,
    0.93, 0.86, 0.80, 0.74, 0.68, 0.62, 0.56, 0.50])
N_m = []
for i in range(len(mass_bin) - 1):
    N_m.append(N_r[i]/(mass_bin[i] - mass_bin[i+1])) #considering the mass-to-light
        ratio.
N_m.append(N_r[-1]/0.06)# The number density of stars

t_sun = 1 * 10**10 #lifetime of sun
#consider lifetime of stellar is propotional to M^(-3)
#Assume that SFR is constant, lifetime of star whose mass is k*M_sun
#The density of stars with mass at k*M_sun is SFR*t(k*M_sun) = N_now, which N_now is
    number density
#The total density of stars is SFR*T0 = N_tot
#N_tot/N_now = T0/t(k*M_sun)=(M_sun)^(-3)/(k*M_sun)^(-3) = (1/k)^(-3)
plt.figure(figsize=(16,9))
N_tot = N_m * (1/10**(mass_bin - 1))**(-3)
phi = np.log10(N_m)+10 + 3*(mass_bin-1)
D = np.linspace(1,100,30)
M = np.linspace(-1,0.5,30)
N1 = []
for i in range(len(M)-1):
    if M[i]<-0.4:
        dat = result[(result['Mass']>M[i])&(result['Mass']<M[i+1])]
        dat = dat[dat['distance']<(1+M[i]+0.4)*100]
#    m = M[i]
#    if M[i]<0:
#        dat = result[(result['mass']>M[i])&(result['mass']<M[i+1])]
#        dat = dat[dat['distance']<10**(2*M[i]-2*0)*100]
    if M[i]>=-0.4:
        dat = result[(result['Mass']>M[i])&(result['Mass']<M[i+1])]
    n = len(dat)
    dmax = dat['distance'].max()
```

```
    N1.append(np.log10(3*n/4/np.pi/dmax**3/(M[i+1]-M[i])))
for i in range(len(M)-1):
    if M[i]>1/3*np.log10(2):
        N1[i]=N1[i]+3*M[i]-np.log10(2)

D = np.linspace(1,100,30)
M = np.linspace(-1,0.5,30)
N2 = []
for i in range(len(M)-1):
    dat = result[(result['Mass']>M[i])&(result['Mass']<M[i+1])]
    n = len(dat)
    dmax = dat['distance'].max()
    N2.append(np.log10(3*n/4/np.pi/dmax**3/(M[i+1]-M[i])))

plt.figure(figsize=(16,9))
plt.plot(M[:-1],N1,'b',label = 'Initial Mass Function
    Correction',linewidth=4,alpha=0.8)
plt.plot(M[:-1],N2,'r',label = 'Initial Mass Function Origin',linewidth=4,alpha=0.8)
plt.plot(mass_bin[9:] - 1, np.log10(N_m[9:]),'green',linewidth=4,alpha=0.8)
plt.plot(mass_bin[0:10]-1, np.log10(N_tot)[0:10], 'green',label='IMF from
    Hipparcos',linewidth=4,alpha=0.8)
#plt.plot(M,Salpter,label = 'IMF Salpter')
#plt.plot(M[:-1],N_c,label = 'IMF correction')
plt.xlabel(r'Mass $Log[M/M_\odot]$',fontsize=18)
plt.ylabel(r'Number Density $Log[\phi]$',fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.legend(loc='upper right', fontsize=18)
plt.title('Initial Mass Function',fontsize=30)
#plt.xlim(-1,0.5)
#plt.ylim(-4,0)
plt.grid()
```