Equations used in Code

Amandeep Singh
Supervised by Prof. Dr. Romain Teyssier

Nov 20th, 2018

$$
\begin{aligned}
n_H &= \mathit{logspace}(-4, 5, 100) && (1)\\
x &= \mathit{log}(\frac{n_H}{n_{H,mean}}) && (2)\\
pdf &= \frac{1}{\sqrt{2\pi (vel_{disp})^2}} \quad e^{[\frac{(-1)}{2}(\frac{x-x_{mean}}{(vel_{disp})})^2]} && (3)\\
(vel_{disp}) &= \sqrt{\log\left(1 + [0.3 \cdot Mach_{no}]^2\right)} && (4)\\
\lambda_{Jeans} &= \frac{c_s}{(\sqrt{4\pi \; G \; n_H \; m_p})} && (5)\\
\tau &= \kappa \; n_H \; \lambda_{Jeans} && (6)\\
\kappa &= \kappa_{dust} \; m_p \qquad ; \qquad \kappa_{dust} = 10^3 && (7)\\
n_{LW} &= (n_{LW})_{out} \; e^{-\tau} \qquad ; \qquad (n_{LW})_{out} = 1 && (8)\\
\chi_{H_2} &= \frac{1}{2 + [\frac{(DC) \; (n_{LW})}{(CC) \; Z \; n_H}]} && (9)\\
& \mathit{[DC = destruction\ coeff.} = 1.7 * 10^{-11}\mathit{]}; && (10)\\
& \mathit{[CC = creation\ coeff.} = 2.5 * 10^{-17}\mathit{]} && (11)\\
n_{H_2} &= n_H \; \chi_{H_2} && (12)
\end{aligned}
$$

Till here, Sir, everything is fine; I am getting the plots between χ_{H_2} and $\log(n_H)$ as expected. The plots obtained are attached for your ready reference:

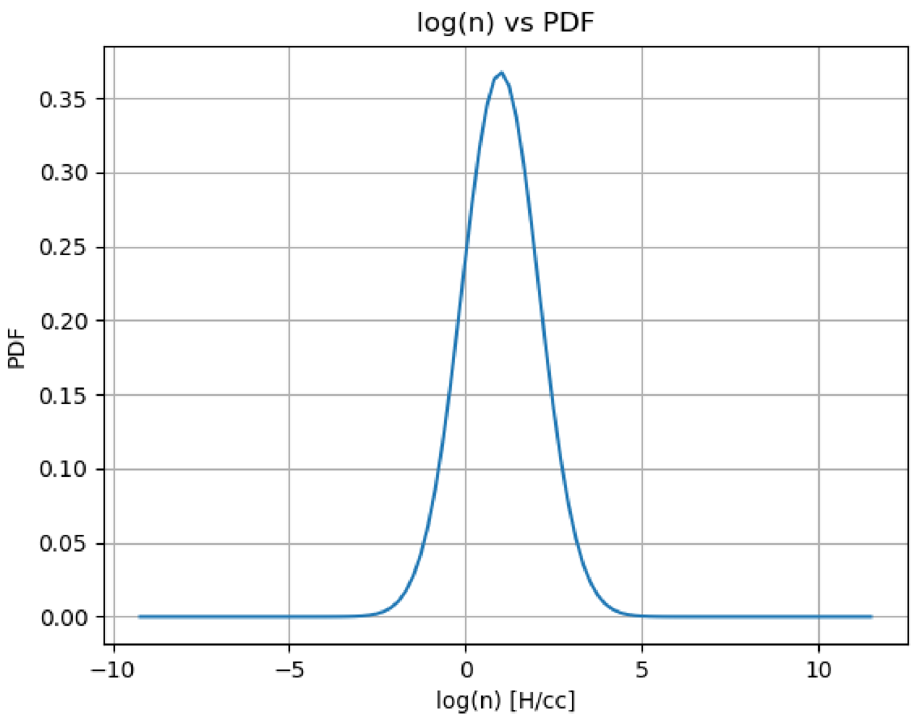


Figure 1: $log(n_H)$ vs PDF

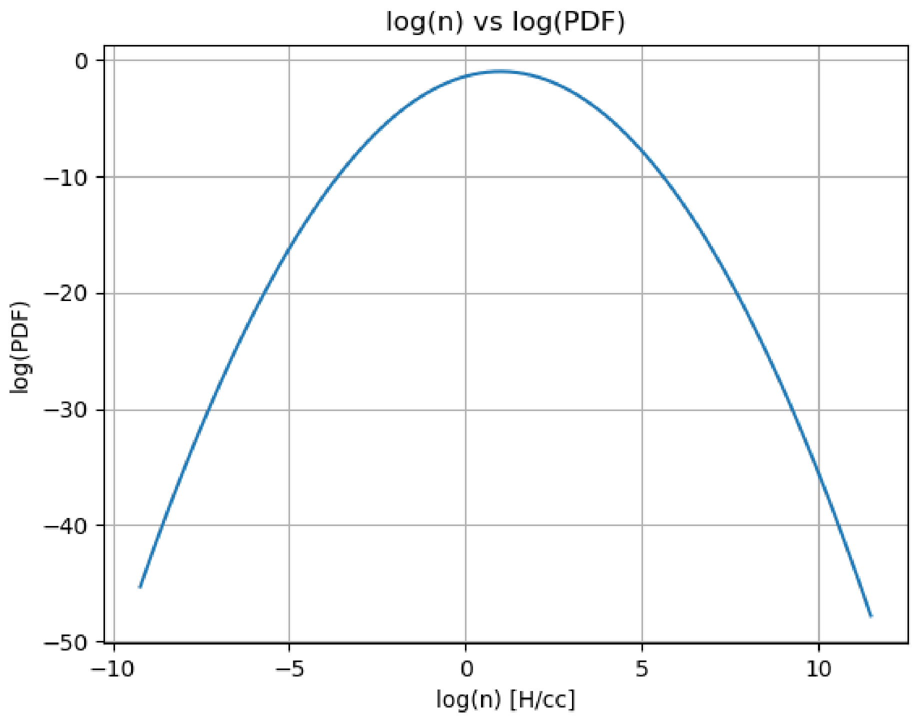


Figure 2: $log(n_H)$ vs $log(PDF)$

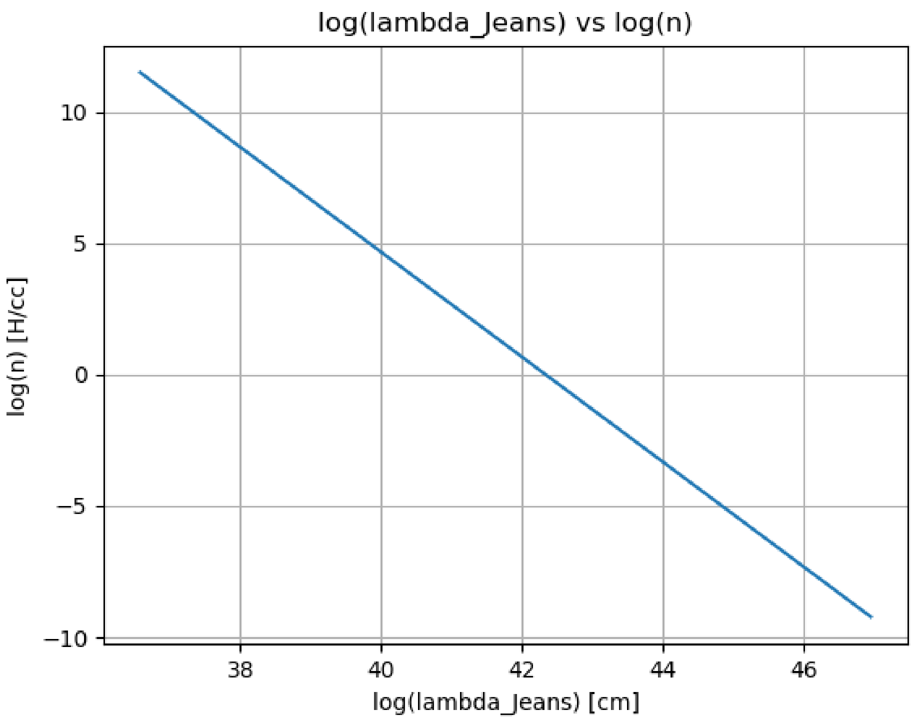


Figure 3: $log(\lambda_{Jeans})$ vs $log(n_H)$

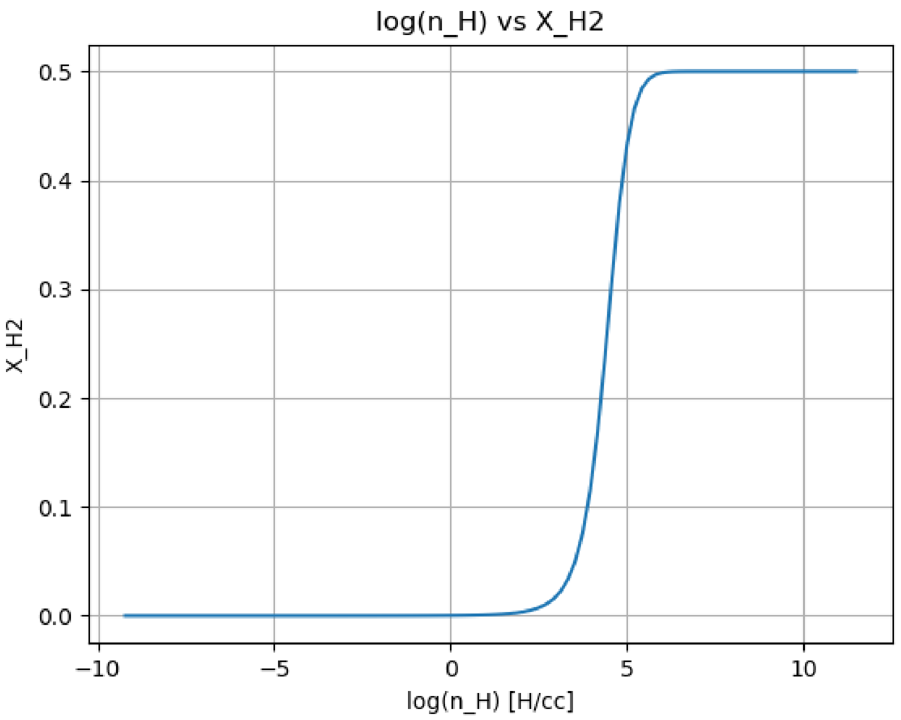


Figure 4: $log(n_H)$ vs χ_{H_2}

$$\bar{n}_{H,mean} = \int [n_H \cdot pdf] \tag{13}$$

$$\bar{n}_{H_2} = \int [n_H \cdot pdf \cdot \chi_{H_2}] \tag{14}$$

$$\bar{\chi}_{H_2} = \frac{\bar{n}_{H,mean}}{\bar{n}_{H_2}} \tag{15}$$

After this I had to find a way to integrate the product of n_H and pdf from the minimum value of n_H to the maximum value of n_H.

My first choice was to use the simple Trapezoidal rule (divide the area under the curve into many rectangles, and then summing the areas of those rectangles), but then I found out that Python already has a built in function for integration, called scipy.integrate.quad() . I assumed that this would be more accurate than the Trapezoidal rule, because they would also take into account the slope of the curved line between the rectangles and then take the half-point to calculate that area as well, thus reducing the error. I tested this hypothesis by making a simple function that returned the square of a variable x, and integrated that function, while I used the same function to calculate the integral using the Trapezoidal rule. I found that this built-in quad function was accurate upto 15 decimal places, and computing time was in micro-seconds, whereas the other method achieved a similar level of precision (11 decimal places) in 2 seconds. So my obvious choice was using this built-in quad function. On a side-note, I looked into the documentation of this function, and found that this was based on a technique from the Fortran library QUADPACK.

The rest of the difficulty was to figure out how this function works. It takes in 4 arguments essentially: (1) the *function* that needs to be integrated, (2) the lower limit of integration, (3) the upper limit of integration, and (4) the arguments that are needed for the *function* in (1).

I was able to set the limits from $-\infty$ to ∞ , and also set the other requirements for this quad function. The plot I obtained is attached for your ready reference.

Clearly, there is some bug in the code. I have seen the documentation of quad(), and have gone though every question posted on StackOverflow related to this quad(), which helped me improve on this plot, but it is still not what we want. I am also attaching the code-snippet for your kind perusal.

While I was trying to figure out quad() on one front, I was also trying to plot the behaviour of these quantities when $n_{H,mean}$, Z , and $Mach_{no}$ are varied, as you desired. I will also attach those plots for your kind perusal.

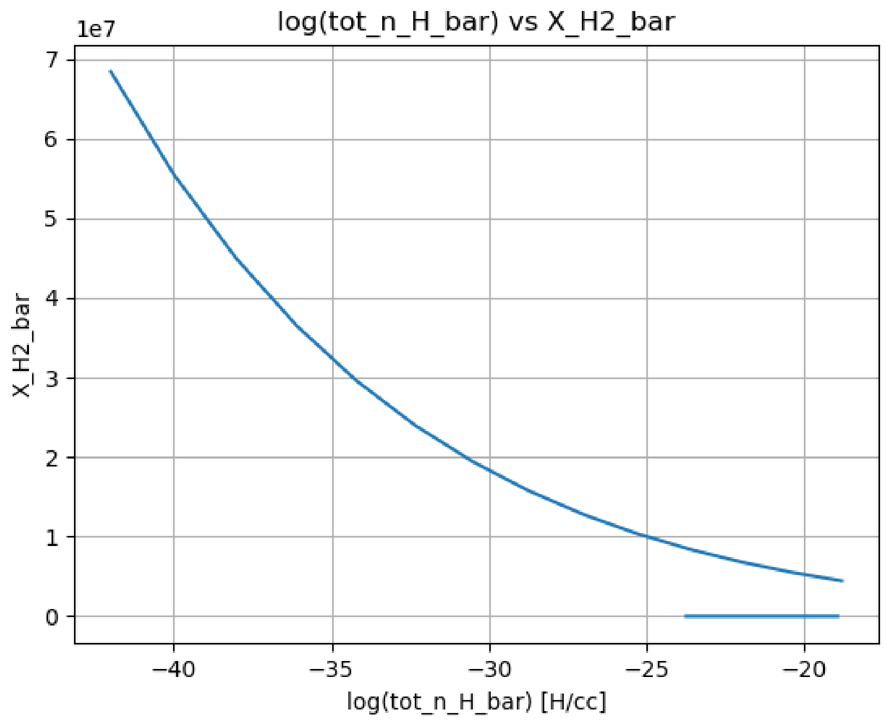
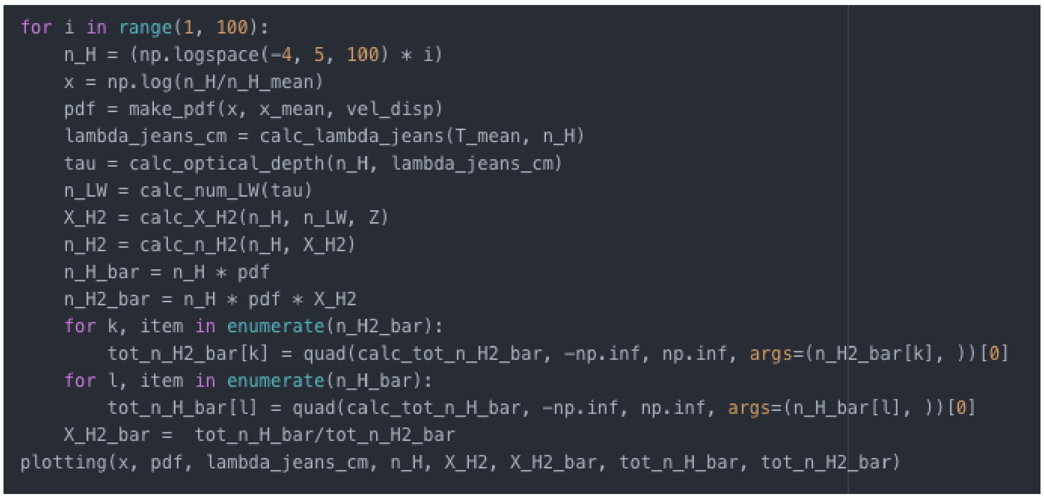


Figure 5: $log(\bar{n}_{H,mean})$ vs $\bar{\chi}_{H_2}$



```
for i in range(1, 100):
    n_H = (np.logspace(-4, 5, 100) * i)
    x = np.log(n_H/n_H_mean)
    pdf = make_pdf(x, x_mean, vel_disp)
    lambda_jeans_cm = calc_lambda_jeans(T_mean, n_H)
    tau = calc_optical_depth(n_H, lambda_jeans_cm)
    n_LW = calc_num_LW(tau)
    X_H2 = calc_X_H2(n_H, n_LW, Z)
    n_H2 = calc_n_H2(n_H, X_H2)
    n_H_bar = n_H * pdf
    n_H2_bar = n_H * pdf * X_H2
    for k, item in enumerate(n_H2_bar):
        tot_n_H2_bar[k] = quad(calc_tot_n_H2_bar, -np.inf, np.inf, args=(n_H2_bar[k], ))[0]
    for l, item in enumerate(n_H_bar):
        tot_n_H_bar[l] = quad(calc_tot_n_H_bar, -np.inf, np.inf, args=(n_H_bar[l], ))[0]
    X_H2_bar =  tot_n_H_bar/tot_n_H2_bar
plotting(x, pdf, lambda_jeans_cm, n_H, X_H2, X_H2_bar, tot_n_H_bar, tot_n_H2_bar)
```

Figure 6: Code snippet

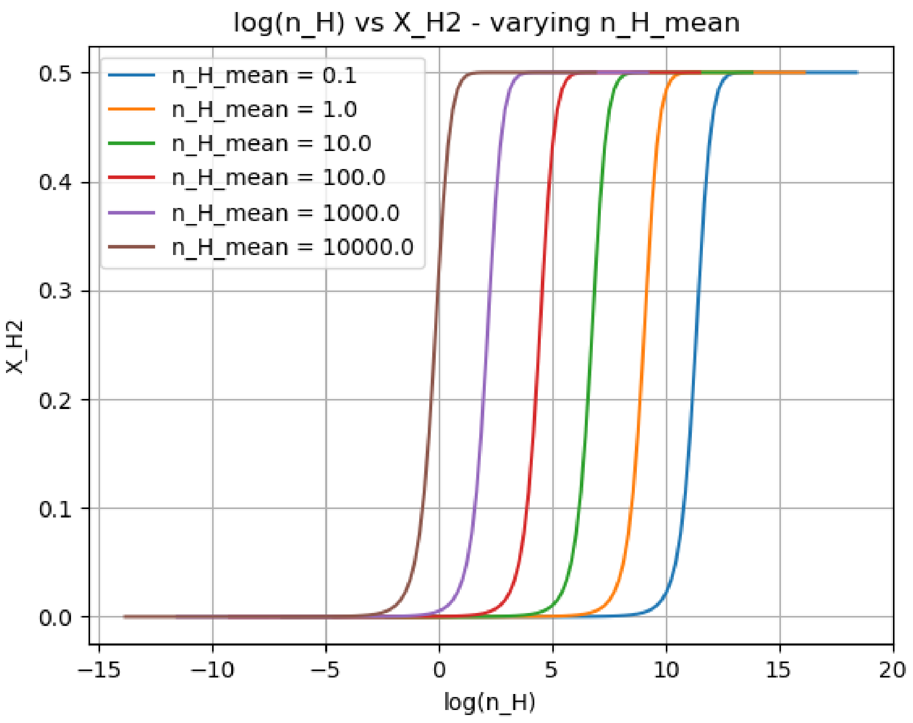


Figure 7: Varying $n_{H,mean}$

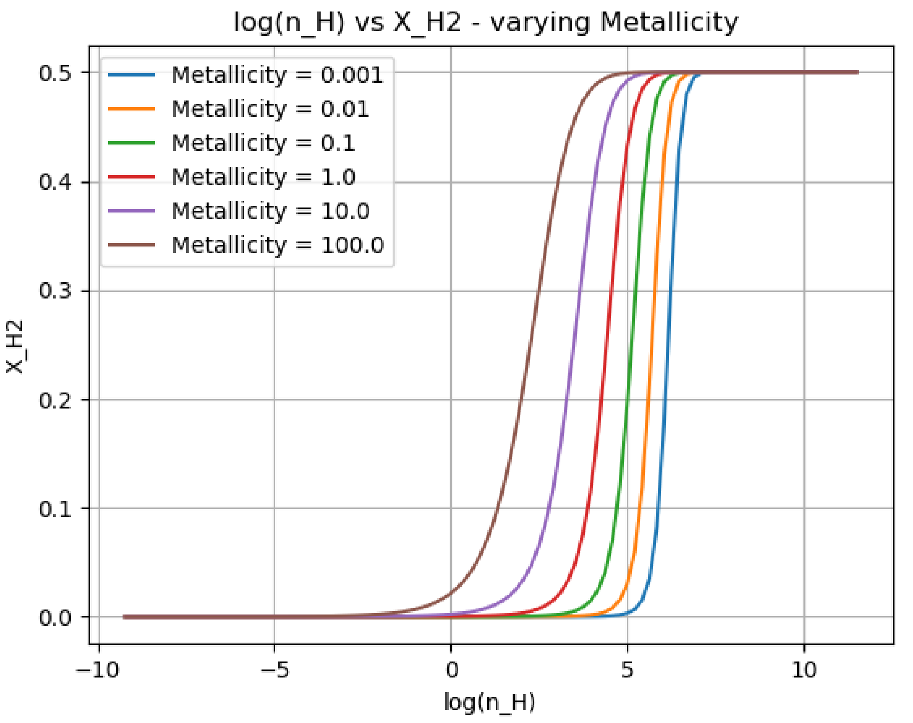


Figure 8: Varying Z

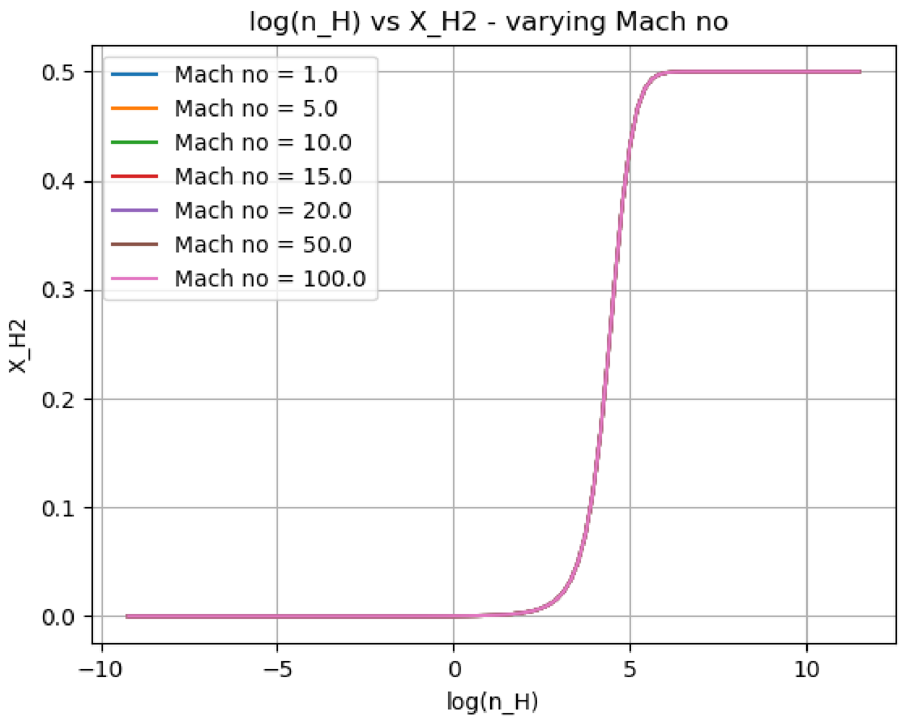


Figure 9: Varying $Mach_{no}$

As can be seen the plot for varying $Mach_{no}$ doesn't show changes for different values of $Mach_{no}$. If you desire, I will plot such plots for all the other values in the code. Currently, I have not included these varying plots for the values that require integration.