Three Exceptions in the Calculation of Relic Abundances

Review of Paper by Kim Griest and David Seckel, 1991

Internship Report

Deepti Hariharan

Advisor: Prof. Manuel Drees

Universität Bonn
13 August, 2020

Abstract

The annihilation and freeze-out mechanisms of elementary particles determine their present-day relic abundance. This is an important tool to study potential dark matter candidates. Relic abundances are usually calculated by the Lee-Weinberg method. The present-day abundance is determined by the annihilation of the particle after it becomes nonrelativistic. Calculation is done using standard approximate solutions of the Boltzmann equation. Kim Griest and David Seckel found three situations in which the standard method of calculating the relic abundances fail [1]. The first situation occurs when another particle's mass is nearly degenerate with the mass of the relic particle. In this case, abundance of the relic is affected by the annihilation of the other particle, called coannihilation. The second situation is when the mass of the relic particle lies near a threshold. This leads to annihilation into channels which were kinematically forbidden. Finally, the third situation occurs when annihilation of the relic particle takes place near a pole in the cross-section. Here, I have reviewed all these situations and attempted to reproduce their results.

Contents

1. Standard Method for Calculation of Relic Abundances

The relic abundance is calculated by finding the number density n of the relic particle χ using the Boltzmann equation

$$\frac{dn}{dt} = -3Hn - \langle\sigma v\rangle(n^2 - n_{eq}^2) \tag{1}$$

where H is the Hubble parameter, $\langle\sigma v\rangle$ is the thermally averaged cross-section with relative velocity v and n_{eq} is the equilibrium number density. The first term in **??** is due to the expansion of the universe and the second term is due to the reactions taking place. The nonrelativistic approximation for the equilibrium number density is $n_{eq} = g(mT/2\pi)^{3/2}exp(-m/T)$ where g is the number of degrees of freedom of the χ particle, m is its mass and T is its temperature. The cross-section σ is Taylor expanded as $\sigma = a + bv^2 \implies \langle\sigma v\rangle = a + 6bT/m$. During freeze-out, the universe was radiation dominated and the time-temperature relation was

$$t = \frac{M_{Pl}}{2 \times 1.66 g_*^{1/2} T^2} = \frac{M_{Pl}}{2 \times 1.66 g_*^{1/2} m^2}\, x^2 \tag{2}$$

where x is the scaled freeze-out temperature ($x = m/T$). Changing the variable from t to x, the Boltzmann equation is

$$\frac{dn}{dx} = -\frac{3n}{x} - \frac{M_{Pl}\,\langle\sigma v\rangle}{1.66 g_*^{1/2} m^2}\, x\,(n^2 - n_{eq}^2) \tag{3}$$

At early times $n \approx n_{eq}$. As the temperature drops below mass of χ, n_{eq} drops exponentially until it reaches freeze-out and the reactions drop out of equilibrium. After this point, the n_{eq} term can be ignored. The solution of eq. (3) is found by solving in these two regimes and equating at the freeze-out point

$$x_f = \ln \frac{0.038 g_\chi M_{Pl} m_\chi \langle\sigma v\rangle}{g_*^{1/2} x_f^{1/2}} \tag{4}$$

where g_* is the total number of effective relativistic degrees of freedom and M_{Pl} is the Planck mass. x_f can be found by numerically solving eq. (4). For candidates of cold dark matter, the freeze-out temperature is of order $m/25$.

At freeze-out, the number density of the relic particle is approximately equal to its equilibrium number density. There is still significant annihilation of the particle after freeze-out which has to

be considered to calculate its present-day density. The abundance of the relic particle is given by

$$\Omega h^2 = \frac{1.07 \times 10^9}{J g_*^{1/2} M_{Pl}} \text{ GeV}^{-1} \tag{5}$$

where Ω is the present-day mass density divided by the critical density, h is the Hubble parameter and J is the efficiency of the annihilation after freeze-out given by

$$J = \int_{x_f}^{\infty} \frac{\langle \sigma v \rangle}{x^2} \, dx \tag{6}$$

Then, for the Taylor expanded cross-section

$$\Omega h^2 = \frac{1.07 \times 10^9 x_f}{g_*^{1/2} M_{Pl} (a + 3b/x_f)} \text{ GeV}^{-1} \tag{7}$$

This method works well for cross-sections which are not too sensitive to temperature. But for the cases discussed in the paper, the cross-section has stronger dependence on temperature and it had to be modified accordingly.

Comparison of standard method and numerical method

To check that this standard method is a good approximation, the Boltzmann equation can be evaluated numerically and compared.

For a given value of x_f, the cross-section can be calculated using eq. (4). The abundance from the standard method is calculated using eq. (5), where now

$$J = \langle \sigma v \rangle \int_{x_f}^{\infty} \frac{1}{x^2} \, dx = \frac{\langle \sigma v \rangle}{x_f} \tag{8}$$

The Boltzmann equation [eq. (3)] equation is numerically evaluated using the fourth order Runge-Kutta method for the number density. The initial values are set to $x_0 = x_f$ and $n_0 = n_{eq} = g_1 m_1^3/(2\pi x_0)^{3/2} \exp(-x_0)$. The step size used is small and dependent on x. Iteration is done for a final value of x derived by assuming that temperature today is of $O(1)$ K $\approx O(10^{-13})$ GeV and $x_{today} = m/T_{today} \sim O(10^{16})$ for $m = 1000$ GeV. I have ignored the change in the temperature-time dependence when the universe moves into a different epoch for this calculation. The abundance is related to the number density by

$$\Omega h^2 = \frac{m_\chi n h^2}{\rho_c^2} \tag{9}$$

as eq. (5) gives the present-day mass density divided by the critical density ρ_c. The comparison of numerical values with the standard method is shown in fig. 1 for varying x_f. The y-axis is scaled logarithmically. To check the independence of the initial values, x_0 is set to $x_f - 5$ and the same procedure is done. The abundances from two initial values differ only by about 0.5%. I did the coding using Python, it is given in appendix A. The numerical values are a decent match with the standard method with an error of about 4%.

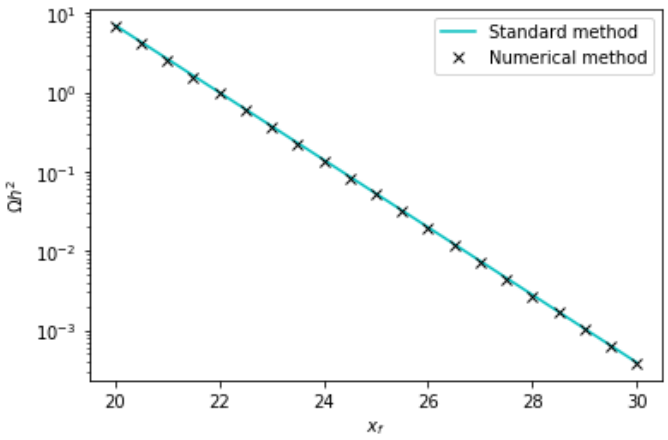


Figure 1: Comparison of the numerical Boltzmann equation and the standard approximation

2. Coannihilation

Coannihilation occurs if the relic particle's mass is nearly degenerate with the mass of other particles. The heavier particles decay into the lighter relic and change its abundance. For example, in supersymmetry, the relic χ_i would correspond to the lightest supersymmetric particle (LSP) and the heavier particles would be squarks. This situation is significant only if the mass difference $\delta m \approx T_f$.

Cross-section

Abundance of the χ_i is determined using the Boltzmann equation

$$\begin{aligned} \frac{dn}{dt} &= -3Hn - \langle \sigma_{eff} v \rangle \left(n^2 - n_{eq}^2\right) \\ \implies \frac{dn}{dx} &= -\frac{3n}{x} - \frac{M_{Pl} \langle \sigma_{eff} v \rangle}{1.66 g_*^{1/2} m^2} \, x \left(n^2 - n_{eq}^2\right) \end{aligned} \tag{10}$$

where

$$\sigma_{eff} = \sum_{ij}^{N} \sigma_{ij} \frac{g_i g_j}{g_{eff}^2} (1 + \Delta_i)^{3/2} (1 + \Delta_j)^{3/2} \exp[-x(\Delta_i + \Delta_j)] \tag{11}$$

with $g_{eff} = \sum_{i=1}^{N} g_i (1 + \Delta_i)^{3/2} \exp(-x\Delta_i)$ and $\Delta_i = (m_i - m_1)/m_1$

In the case of supersymmetry, it is expected that $\sigma_{22} \approx (\alpha_s/\alpha)\sigma_{12} \approx (\alpha_s/\alpha)^2 \sigma_{11}$ for a two-particle system where α is the electroweak coupling and α_s is the strong coupling. The effective cross-section is then

$$\begin{aligned} \sigma_{eff} &= \sigma_{11} \frac{g_1^2}{g_{eff}^2} [(1 + \Delta_1)^{3/2}]^2 \exp[-2x\Delta_1] + 2\sigma_{12} \frac{g_1 g_2}{g_{eff}^2} (1 + \Delta_1)^{3/2} (1 + \Delta_2)^{3/2} \exp[-x(\Delta_1 + \Delta_2)] \\ &\qquad + \sigma_{22} \frac{g_2^2}{g_{eff}^2} [(1 + \Delta_2)^{3/2}]^2 \exp[-2x\Delta_2] \\ &= \sigma_{11} \frac{g_1^2}{g_{eff}^2} [(1 + \Delta_1)^{3/2}]^2 \exp[-2x\Delta_1] + 2A\sigma_{11} \frac{g_1^2}{g_{eff}^2} \frac{g_2}{g_1} (1 + \Delta_1)^{3/2} (1 + \Delta_2)^{3/2} \exp[-x(\Delta_1 + \Delta_2)] \\ &\qquad + A^2 \sigma_{11} \frac{g_1^2}{g_{eff}^2} \frac{g_2^2}{g_1^2} [(1 + \Delta_2)^{3/2}]^2 \exp[-2x\Delta_2] \end{aligned}$$

where $A = (\alpha_s/\alpha) \approx 20$ is the enhancement. Defining $w = (1+\Delta)^{3/2}\exp(-x\Delta)g_2/g_1$, the effective number of degrees of freedom is $g_{eff} = g_1(1+w)$ and

$$\sigma_{eff} = \sigma_{11}\left(\frac{1+Aw}{1+w}\right)^2 \tag{12}$$

For a single squark, $g_2/g_1 = 3$ and for all six flavors $g_2/g_1 = 18$.

Annihilation integral and abundance

The modified scaled freeze-out temperature is given by

$$\begin{aligned} x_f &= \ln \frac{0.038 g_{eff} M_{Pl} m_1 \langle \sigma_{eff} v \rangle}{g_*^{1/2} x_f^{1/2}} \\ &= \ln \frac{0.038 g_1 M_{Pl} m_1 \langle \sigma_{11} v \rangle}{g_*^{1/2} x_f^{1/2}} \frac{(1+Aw)^2}{1+w} \end{aligned} \tag{13}$$

and the annihilation integral is

$$J = \frac{a_{11} I_a + 3 b_{11} I_b / x_f}{x_f} \tag{14}$$

where

$$\begin{aligned} I_a &= \frac{x_f}{a_{11}} \int_{x_f}^{\infty} \frac{a_{eff}}{x^2}\, dx \\ I_b &= \frac{2x_f^2}{b_{11}} \int_{x_f}^{\infty} \frac{b_{eff}}{x^3}\, dx \end{aligned} \tag{15}$$

The coefficients a_{eff} and b_{eff} are given by similar formulas as eq. (12)

$$a_{eff} = a_{11}\left(\frac{1+Aw}{1+w}\right)^2 \qquad\qquad b_{eff} = b_{11}\left(\frac{1+Aw}{1+w}\right)^2 \tag{16}$$

The relic abundance is calculated using the new freeze-out temperature [eq. (13)] and is given by

$$\Omega h^2 = \frac{1.07 \times 10^9 x_f}{g_*^{1/2} M_{Pl} (a_{11} I_a + 3 b_{11} I_b / x_f)} \tag{17}$$

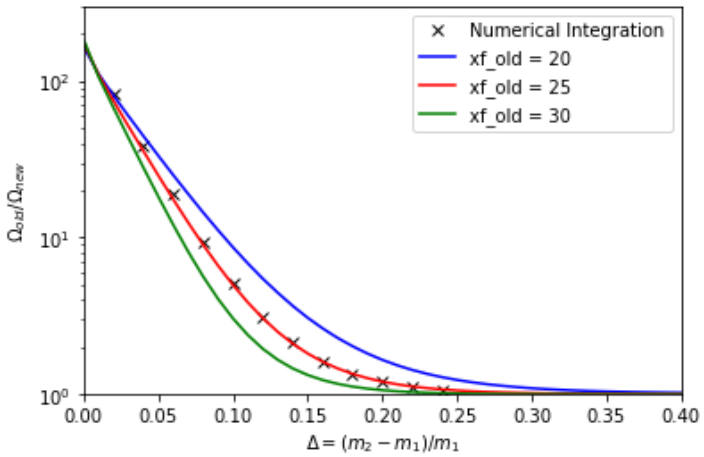
The abundances calculated using the old and new methods are compared by the ratio

$$\begin{aligned} R &= \frac{\Omega_{old}}{\Omega_{new}} = \frac{J_{new}}{J_{old}} \\ &= \frac{\int_{x_{f,new}}^{\infty} \frac{a_{eff}}{x^2}\, dx + 6\int_{x_{f,new}}^{\infty} \frac{b_{eff}}{x^3}\, dx}{a + 6b/x_{f,old}} \\ &= \frac{\int_{x_{f,new}}^{\infty} \frac{a_{11}}{x^2}\left(\frac{1+A(1+\Delta)^{3/2}\exp(-x\Delta)}{1+(1+\Delta)^{3/2}\exp(-x\Delta)}\right)^2 dx + 6\int_{x_{f,new}}^{\infty} \frac{b_{11}}{x^3}\left(\frac{1+A(1+\Delta)^{3/2}\exp(-x\Delta)}{1+(1+\Delta)^{3/2}\exp(-x\Delta)}\right)^2 dx}{a + 6b/x_{f,old}} \end{aligned} \tag{18}$$

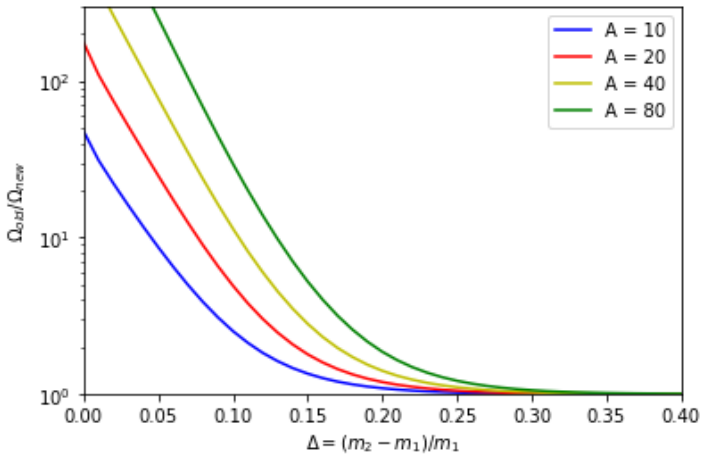
Figure 2a, shows the decrease in relic abundance with ratio of masses. The enhancement A and number of coannihilation species g_2/g_1 was kept constant and the scaled freeze-out temperature $x_{f,old}$ was varied. For each value of $x_{f,old}$, the cross-section $\langle\sigma_{11}v\rangle$ was calculated using eq. (4). I assumed that the cross-section for x_{new} is also the same as x_old. The value of $x_{f,new}$ was obtained iteratively from eq. (13). I have used $a_{11} = b_{11} = a = b$, but in principle, these coefficients must be calculated from the matrix element based on the type of reaction and initial and final particles. The crosses are the result of numerically evaluating the Boltzmann equation using the fourth-order Runge-Kutta method for $x_{f,old} = 25$ using a similar method as discussed in chapter 1. The initial value $x_0 = x_f - 2$ and $n_0 = n_{eq} = g_1 m_1^3/(2\pi x_0)^{3/2} \exp(-x_0)$ was used here. For finding n_{new}, the cross-section has the additional factor $[(1 + Aw)/(1 + w)]^2$ from eq. (12). The numerical curve is a good match with the standard method.
In fig. 2b, x_{f_old} and g_2/g_1 was kept constant and the enhancement A was varied and in fig. 2c, $x_{f,old}$ and A was kept constant and the number of species was varied. A link to the Python codes is given in appendix A.

Conclusion

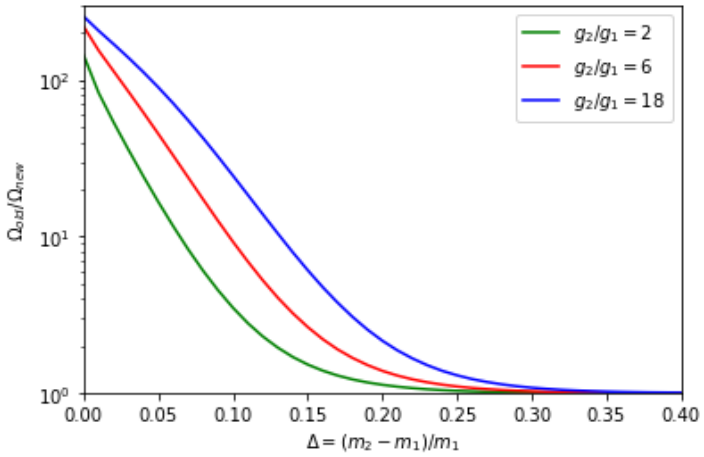
Based on their results, the authors concluded that the relic abundance changes significantly when the heavier particle is within 10% of the lightest particle's mass. For higher values of A and g_2/g_1, mass differences of 15–20% also can lead to significant change in the abundance. My results are comparable to the authors' with a maximum error of about 25%.



(a) $x_{f,old}$ is varied while setting $A = 20$ and $g_2/g_1 = 3$. The crosses indicate numerical evaluation of the Boltzmann equation



(b) A is varied while setting $x_{f,old} = 25$ and $g_2/g_1 = 3$



(c) g_2/g_1 is varied while setting $x_{f,old} = 25$ and $A = 20$

Figure 2: Decrease in relic abundance due to coannihilation

3. Annihilation into Forbidden Channels

For annihilations to happen into forbidden channels, the mass relic particle must be slightly below the mass threshold. The threshold is where there would be a significant amount of annihilations if the particle were more massive. At the relevant freeze-out temperature of $T_f = m_1/25$, the relative velocity is not zero and annihilations into heavier particle may take place. These annihilations dominate if the masses of the final particles are close to that of the relic.

Cross-section

For relativistic particles, the cross-section is Taylor expanded as $(\sigma v) = a + bv^2$. But in the nonrelativistic case, if the mass of the particles is large, the cross-section is expanded as

$$(\sigma v) = (a' + b'v^2)\, \upsilon_2 \tag{19}$$

where υ_2 is the velocity of the final particles in the center-of-mass frame given by

$$\upsilon_2 = z\sqrt{\frac{v^2}{4} + \mu_+^2} \tag{20}$$

where $z = m_2/m_1$ and $\mu_+ = \sqrt{1-z^2}/z$. It has been assumed that both the final state particles has mass m_2 and the relic has mass m_1. Away from the threshold, υ_2 can be Taylor expanded as

$$\begin{aligned} \upsilon_2 &\approx z\left(\mu_+ + \frac{v^2}{8\mu_+}\right) \\ &= \sqrt{1-z^2}\left(1 + \frac{z^2v^2}{8(1-z^2)}\right) \end{aligned} \tag{21}$$

Annihilations may also occur above the threshold in allowed channels and if the mass of the final particles are large, they cannot be ignored. The cross-section for such channels is

$$\langle\sigma v\rangle_{allowed} = \frac{x^{3/2}}{2\sqrt{\pi}}\int_0^\infty dv\, v^2 e^{-v^2x/4}\upsilon_2(a' + b'v^2) \tag{22}$$

Changing the integration variable into $t = v^2x/4 \implies dv = dt/\sqrt{xt}$ and using $\upsilon_2 = z\sqrt{\frac{v^2}{4} + \mu_+^2} = z\sqrt{\frac{t}{x} + \mu_+^2}$, the cross-section becomes

$$\langle\sigma v\rangle_{all} = \frac{2z}{\sqrt{\pi}}\int_0^\infty \sqrt{\frac{t^2}{x} + \mu_+^2 t}\,\left(a' + \frac{4b't}{x}\right)e^{-t}dt \tag{23}$$

In case of the forbidden channel processes, the reactions take place only after a critical velocity $v_c = 2\mu_- = 2\sqrt{z^2-1}/z$ where they are activated. The cross-section for forbidden channels is

$$\langle\sigma v\rangle_{forbidden} = \frac{x^{3/2}}{2\sqrt{\pi}} \int_{v_c}^{\infty} dv\, v^2 e^{-v^2 x/4}\, \upsilon_2\, (a' + b' v^2) \tag{24}$$

Changing the integration variable to $t = v^2 x/4 - \mu_-^2 x \implies dv = dt/\sqrt{x(t^2+\mu_+^2 x)}$ and using eq. (20) with μ_- instead of μ_+; $\upsilon_2 = z\sqrt{\frac{v^2}{4} - \mu_-^2} = z\sqrt{\frac{t}{x}}$, the cross-section becomes

$$\langle\sigma v\rangle_{for} = e^{-\mu_-^2 x} \frac{2z}{\sqrt{\pi}} \int_0^{\infty} \sqrt{\frac{t^2}{x} + \mu_-^2 t}\, \left((a' + 4b'\mu_-^2) + \frac{4b't}{x}\right) e^{-t} dt \tag{25}$$

At the threshold, $z = 1$ which means $\mu_+ = \mu_- = 0$ and the cross-section for the allowed and forbidden channels are equal

$$\begin{aligned}\langle\sigma v\rangle_{thr} &= \frac{2z}{\sqrt{\pi}} \int_0^{\infty} \frac{t}{\sqrt{x}} e^{-t} \left(a' + \frac{4b't}{x}\right) dt \\ &= \frac{2}{\sqrt{\pi x}} \left(a' + \frac{8b'}{x}\right)\end{aligned} \tag{26}$$

The cross-section can be solved analytically for s-wave annihilations. For the case of forbidden channel, the s-wave cross-section is

$$\begin{aligned}\langle a'\upsilon_2\rangle_{for} &= e^{-\mu_-^2 x} \frac{2z}{\sqrt{\pi}} \int_0^{\infty} \sqrt{\frac{t^2}{x} + \mu_-^2 t}\; a'\; e^{-t} dt \\ &= a' e^{-\mu_-^2 x} \frac{2z}{\sqrt{\pi}} \frac{1}{2} e^{\mu_-^2 x/2} \frac{1}{\sqrt{\mu_-^2 x}} (\mu_-^2 x)^{3/2} K_1(\mu_-^2 x/2) \;\; \text{for } \mathrm{Re}(\mu_-^2 x) > 0 \\ &= a' \mu_-^2 x \sqrt{\frac{x}{\pi}}\; e^{-\mu_-^2 x/2} K_1(\mu_-^2 x/2)\end{aligned} \tag{27}$$

where K_1 is the modified Bessel function. Similarly the s-wave cross-section of the allowed channel is

$$\begin{aligned}\langle a'\upsilon_2\rangle_{all} &= \frac{2z}{\sqrt{\pi}} \int_0^{\infty} \sqrt{\frac{t^2}{x} + \mu_+^2 t}\; a'\; e^{-t} dt \\ &= a' \frac{2z}{\sqrt{\pi}} \frac{1}{2} e^{\mu_+^2 x/2} \frac{1}{\sqrt{\mu_+^2 x}} (\mu_+^2 x)^{3/2} K_1(\mu_+^2 x/2) \;\; \text{for } \mathrm{Re}(\mu_+^2 x) > 0 \\ &= a' \mu_+^2 x \sqrt{\frac{x}{\pi}}\; e^{\mu_+^2 x/2} K_1(\mu_+^2 x/2)\end{aligned} \tag{28}$$

Away from the threshold as $\mu_\pm^2 \to \infty$, K_1 can be approximated as $K_1(\mu_\pm^2 x/2) \approx \sqrt{\pi/(\mu_\pm^2 x)}\; e^{-\mu_\pm^2 x/2}$ and at the threshold $K_1(\mu_\pm^2 x/2) \approx 2/(\mu_\pm^2 x)$. Thus,

$$\langle a'\upsilon_2\rangle = \begin{cases} a'\mu_+ z = a'\sqrt{1-z^2}, & z \to 0 \text{ (allowed)} \\ 2a'/\sqrt{\pi x}, & z \to 1 \text{ (threshold)} \\ a'\mu_- z e^{-\mu_-^2 x} = a'\sqrt{z^2-1} e^{-(z^2-1)x/z}, & z \to \infty \text{ (forbidden)} \end{cases} \tag{29}$$

However, the above approximations cannot be used to perform the annihilation integrals. To do so, the cross-section is asymptotically expanded for $\mu_+^2 x \to \infty$. For the allowed case, using the expansion

$$\sqrt{\frac{t^2}{x} + \mu_+^2 t} = \frac{t}{\sqrt{x}} \sqrt{1 + \frac{\mu_+^2 x}{t}} \approx \mu_+ \sqrt{t} \left(1 + \frac{t}{2\mu_+^2 x}\right)$$

the cross-section eq. (23) is

$$\begin{aligned}
\langle \sigma v \rangle_{all} &\approx \frac{2z\mu_+}{\sqrt{\pi}} \int_0^\infty \left(t^{1/2} + \frac{t^{3/2}}{2\mu_+^2 x}\right) a' \, e^{-t} \, dt \; + \; \frac{2z\mu_+}{\sqrt{\pi}} \int_0^\infty \left(t^{3/2} + \frac{t^{5/2}}{2\mu_+^2 x}\right) \frac{4b'}{x} \, e^{-t} dt \\
&= z\mu_+ \left[\, a' \left(1 + \frac{3}{4x\mu_+^2}\right) + \frac{6b'}{x}\left(1 + \frac{5}{4x\mu_+^2}\right) \right] \\
&= \sqrt{1 - z^2} \left[\, a' \left(1 + \frac{3z^2}{4x(1 - z^2)}\right) + \frac{6b'}{x}\left(1 + \frac{5}{4x(1 - z^2)}\right) \right]
\end{aligned} \tag{30}$$

At the threshold, expand in $\mu_\pm x \to 0$ (both eq. (23) and (25) works)

$$\sqrt{\frac{t^2}{x} + \mu_\pm^2 t} = \frac{t}{\sqrt{x}} \sqrt{1 + \frac{\mu_\pm^2 x}{t}} \approx \frac{t}{\sqrt{x}}\left(1 + \frac{\mu_+^2 x}{2t}\right)$$

The cross-section is then (here, $z + 1 \approx 2$ for $z \to 1$)

$$\langle \sigma v \rangle_{thr} \approx \frac{2z}{\sqrt{\pi x}} \left[\, a' \left(1 - x(z - 1)\right) + \frac{8b'}{x}\left(1 - \frac{x(z - 1)}{2}\right) \right] \tag{31}$$

For the forbidden region, the cross-section is asymptotically expanded for $\mu_-^2 x \to \infty$. Using the expansion

$$\sqrt{\frac{t^2}{x} + \mu_-^2 t} = \frac{t}{\sqrt{x}} \sqrt{1 + \frac{\mu_-^2 x}{t}} \approx \mu_- \sqrt{t} \left(1 + \frac{\mu_-^2 x}{2t}\right)$$

the cross-section eq. (25) is

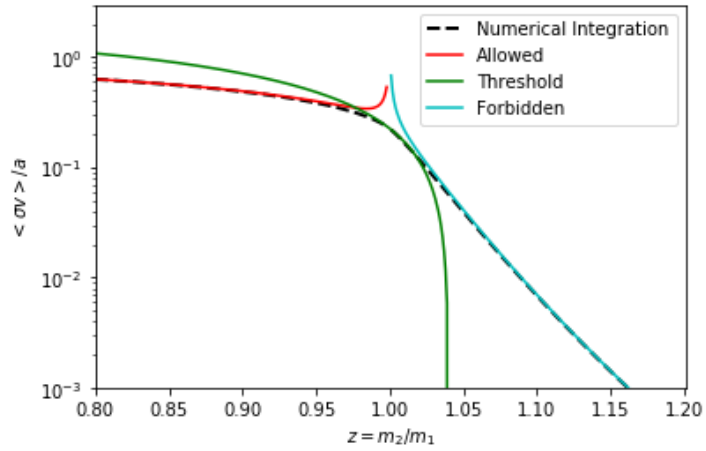
$$\begin{aligned}
\langle \sigma v \rangle_{for} &= e^{-\mu_-^2 x} \frac{2z\mu_-}{\sqrt{\pi}} \int_0^\infty \left(t^{1/2} + \frac{\mu_-^2 x}{2t^{1/2}}\right) \left((a' + 4b'\mu_-^2) + \frac{4b't}{x}\right) e^{-t} dt \\
&\approx z\mu_- e^{-\mu_-^2 x} \left[\, a' \left(1 + \frac{3}{4x\mu_-^2}\right) + 4b'\mu_-^2 \left(1 + \frac{9}{4x\mu_-^2} + \frac{45}{32x^2\mu_-^4}\right) \right] \\
&= \sqrt{z^2 - 1} \, e^{-(z^2-1)x/z^2} \left[\, a' \left(1 + \frac{3z^2}{4x(z^2 - 1)}\right) + 4b' \left(1 + \frac{9}{4x} + \frac{45z^2}{32x^2(1 - z^2)}\right) \right]
\end{aligned} \tag{32}$$

The cross-sections are shown in fig. 3. The dotted line show the numerical integrations eq. (23), (24) and (25) and the solid lines show the analytic approximations eq. (30), (32) and (31) for $x = 25$.

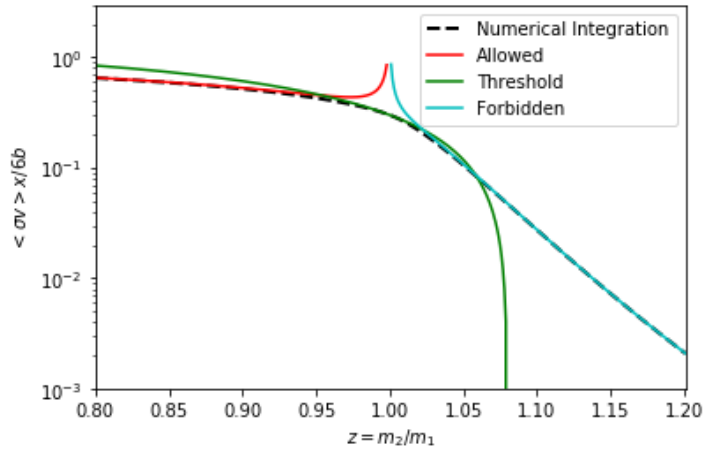
Annihilation integral and abundance

To find the abundance, the annihilation integral

$$J = \frac{a_{tot} + 3b_{tot}/x_f}{x_f} \tag{33}$$



(a) Pure s-wave annihilation



(b) Pure p-wave annihilation

Figure 3: Cross-section for annihilation into forbidden channels

must be performed. Now, x_f is the new value of freeze-out temperature, $a_{tot} = a_{11} + a' I'_a$ and $b_{tot} = b_{11} + b' I'_b$. The coefficients a_{11} and b_{11} correspond to the ordinary cross-section and a' and b' correspond to the forbidden channels. I'_a and I'_b are given by

$$\begin{aligned} I'_a &= \frac{x_f}{a'} \int_{x_f}^{\infty} \frac{\langle a' v_2 \rangle}{x^2}\, dx \\ I'_b &= \frac{2x_f^2}{b'} \int_{x_f}^{\infty} \frac{\langle b' v^2 v_2 \rangle}{6x^2}\, dx \end{aligned} \tag{34}$$

Analytic approximations for I'_a and I'_b are found by substituting eq. (34) in eq. (30), (31) and (32) for the allowed, threshold and forbidden cases respectively.

$$\begin{aligned} I_{a',all} &= \sqrt{1-z^2}\left(1 + \frac{3z^2}{8(1-z^2)x_f}\right) \\ I_{b',all} &= \sqrt{1-z^2}\left(1 + \frac{5z^2}{6(1-z^2)x_f}\right) \end{aligned} \tag{35}$$

$$\begin{aligned} I_{a',thr} &= \frac{4z}{3\sqrt{\pi x_f}}\left(1 - 3x_f(z-1)\right) \\ I_{b',thr} &= \frac{32z}{15\sqrt{\pi x_f}}\left(1 - \frac{5}{6}x_f(z-1)\right) \end{aligned} \tag{36}$$

$$\begin{aligned} I_{a',for} &= \frac{z^2}{\sqrt{z^2-1}x_f} e^{-(z^2-1)x_f/z^2} \\ I_{b',for} &= \frac{4}{3}\sqrt{z^2-1}\, e^{-(z^2-1)x_f/z^2}\left(1 + \frac{z^2}{4(z^2-1)x_f}\right) \end{aligned} \tag{37}$$

For the allowed and threshold regions, the cross-section was directly integrated and for the forbidden region asymptotic expansion was used.

The decrease in relic abundance with respect to z is shown in fig. 4. The y-axis is the ratio

$$\begin{aligned} R = \frac{\Omega_{old}}{\Omega_{new}} &= \frac{J_{new}}{J_{old}} \\ &= \frac{a_{11} + a' I'_a}{x_{f,new}} \frac{x_{f,old}}{a} \\ &= \frac{(1 + A' I'_a) x_{f,old}}{x_{f,new}} \end{aligned} \tag{38}$$

for s-wave annihilations, and

$$\begin{aligned} R = \frac{\Omega_{old}}{\Omega_{new}} &= \frac{J_{new}}{J_{old}} \\ &= \frac{3(b_{11} + b' I'_b)/x_{f,new}}{x_{f,new}} \frac{x_{f,old}}{3b_{11}/x_{f,old}} \\ &= \frac{(1 + A' I'_b) x_{f,old}^2}{x_{f,new}^2} \end{aligned} \tag{39}$$

for $p-$wave annihilations. A' is the factor by which the forbidden channel would dominate the cross-section if it were not suppressed ($a' = A'a_{11}$ and $b' = A'b_{11}$). The dotted lines correspond to numerical integration of I'_a and I'_b [eq. (34)] by substituting the analytic approximations of cross-section [eq. (30), (31), (32)] for $A' = 500$

$$I_{a',all} = \frac{x_f}{a'} \int_{x_f}^{\infty} \frac{1}{x^2} \sqrt{1-z^2}\; a' \left(1 + \frac{3z^2}{4x(1-z^2)}\right) dx$$
$$I_{a',thr} = \frac{x_f}{a'} \int_{x_f}^{\infty} \frac{1}{x^2} \frac{2z}{\sqrt{\pi x}}\; a' \Big(1 - x(z-1)\Big) dx$$
$$I_{a',for} = \frac{x_f}{a'} \int_{x_f}^{\infty} \frac{1}{x^2} \sqrt{z^2-1}\; e^{-(z^2-1)x/z^2}\; a' \left(1 + \frac{3z^2}{4x(z^2-1)}\right) dx$$

$$I_{b'all} = \frac{2x_f^2}{b'} \int_{x_f}^{\infty} \frac{1}{6x^2} \sqrt{1-z^2}\; \frac{6b'}{x} \Big(1 + \frac{5}{4x(1-z^2)}\Big) dx$$
$$I_{b'thr} = \frac{2x_f^2}{b'} \int_{x_f}^{\infty} \frac{1}{6x^2} \frac{2z}{\sqrt{\pi x}}\; \frac{8b'}{x} \Big(1 - \frac{x(z-1)}{2}\Big) dx$$
$$I_{b'for} = \frac{2x_f^2}{b'} \int_{x_f}^{\infty} \frac{1}{6x^2} \sqrt{z^2-1}\; e^{-(z^2-1)x/z^2} 4b' \left(1 + \frac{9}{4x} + \frac{45z^2}{32x^2(1-z^2)}\right) dx$$

The solid lines correspond to the value of $R = \Omega_{old}/\Omega_{new}$ obtained from numerical integration of I'_a and I'_b [eq. (34)] using the result obtained from numerically evaluating the cross-section integrals [eq. (23), (25)] for $A' = 20, 100, 500$

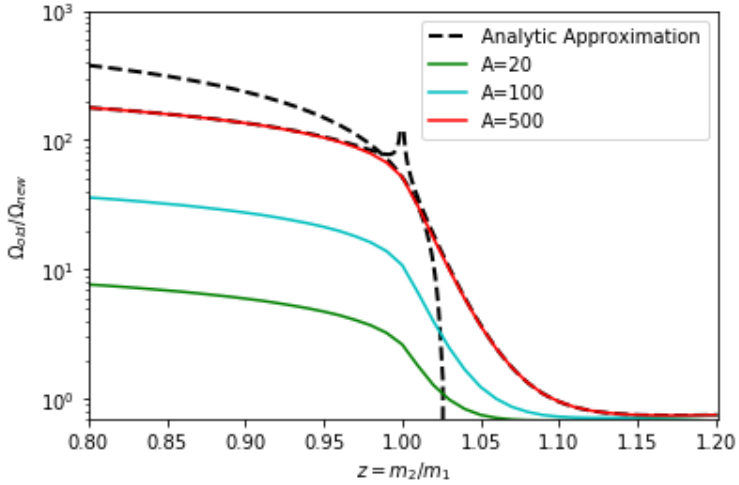
$$I_{a',all} = \frac{x_f}{a'} \int_{x_f}^{\infty} \frac{1}{x^2} \left[\frac{2z}{\sqrt{\pi}} \int_0^{\infty} \sqrt{\frac{t^2}{x} + \mu_+^2 t}\; a'\; e^{-t} dt\right] dx$$
$$I_{a',for} = \frac{x_f}{a'} \int_{x_f}^{\infty} \frac{1}{x^2} \left[e^{-\mu_-^2 x} \frac{2z}{\sqrt{\pi}} \int_0^{\infty} \sqrt{\frac{t^2}{x} + \mu_-^2 t}\; a'\; e^{-t} dt\right] dx$$

$$I_{b',all} = \frac{2x_f^2}{b'} \int_{x_f}^{\infty} \frac{1}{6x^2} \left[\frac{2z}{\sqrt{\pi}} \int_0^{\infty} \sqrt{\frac{t^2}{x} + \mu_+^2 t}\; \frac{4b't}{x}\; e^{-t} dt\right] dx$$
$$I_{b',for} = \frac{2x_f^2}{b'} \int_{x_f}^{\infty} \frac{1}{6x^2} \left[e^{-\mu_-^2 x} \frac{2z}{\sqrt{\pi}} \int_0^{\infty} \sqrt{\frac{t^2}{x} + \mu_-^2 t} \left(4b'\mu_-^2 + \frac{4b't}{x}\right) e^{-t} dt\right] dx$$

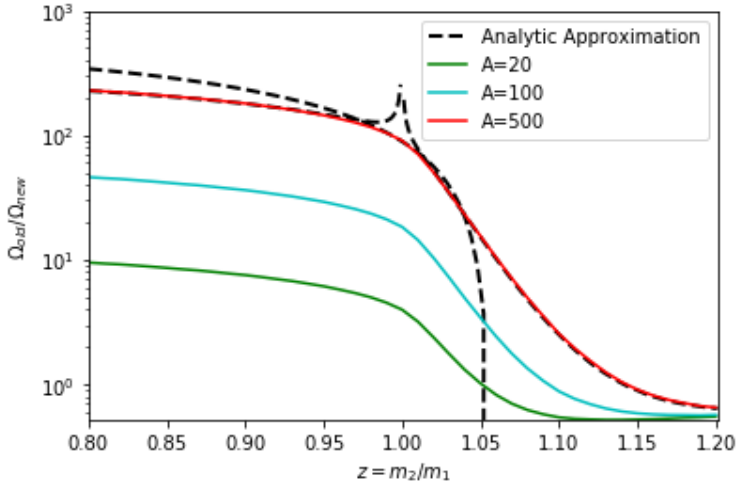
I_{all} is evaluated in the region $z < 1$ and I_{for} is evaluated for $z > 1$. I_{thr} is evaluated for the whole range of z ($0.8 < z < 1.2$). A link to the codes is given in appendix A.

Conclusion

The authors conclude that new channels are important even if the masses are 10–15% below the threshold. The results I got are about 30% lesser in magnitude as compared to theirs.



(a) Pure s-wave



(b) Pure p-wave

Figure 4: Decrease in relic abundance due to annihilation into a forbidden channel

4. Annihilation Near Poles

The situation in which annihilation takes place near a pole in the cross-section is considered in this section. For example, if s-channel exchange of a Z boson happens or the if mass of the relic is about half the mass of a resonance such as the $J/\psi, \eta$ or Υ, there will be a pole in the cross-section.

Cross-section

In the case of a particle such as $Z-$boson being exchanged, the cross-section is

$$\sigma v = \frac{\alpha^2 s}{(M_{ex}^2 - s)^2 + M_{ex}^2 \Gamma_{ex}^2} \tag{40}$$

where $s = 4m_1^2/(1 - v^2/4)$ is the Mandelstam variable, m_1 is the mass of the relic, α is the coupling constant squared, M_{ex} is the mass of the exchange particle and Γ_{ex} is the total width of the exchange particle. Using the scaled variables $u = (2m_1/M_{ex})^2$ and $\epsilon = (\Gamma_{ex}/M_{ex})^2$, the cross-section is

$$\sigma v = \frac{\alpha^2 u/(1 - v^2/4)}{M_{ex}^2\left[\left(1 - u/(1 - v^2/4)\right)^2 + \epsilon\right]} \tag{41}$$

The poles occur at $u = 1 - v^2/4$ where the cross-section is $(\sigma v)_{pole} = \alpha^2/(M_{ex}^2\epsilon)$.

The cross-section is evaluated numerically in the center-of-mass frame by using the formula

$$\begin{aligned}\langle\sigma v\rangle_{num} &= \frac{x^{3/2}}{2\sqrt{\pi}} \int_0^\infty dv\, v^2 (\sigma v)\, e^{-xv^2/4} \\ &= \frac{x^{3/2}}{2\sqrt{\pi}} \int_0^\infty dv\, v^2 \frac{\alpha^2 u/(1 - v^2/4)}{M_{ex}^2\left[\left(1 - u/(1 - v^2/4)\right)^2 + \epsilon\right]}\, e^{-xv^2/4}\end{aligned} \tag{42}$$

The cross-section eq. (41) can also be analytically approximated in various ways. A common method is to Taylor expand it in v^2

$$\begin{aligned}(\sigma v)_{Taylor} &= \frac{\alpha^2 u}{M_{ex}^2} \frac{1 + v^2/4}{[1 - u(1 + v^2/4)]^2 + \epsilon} \\ &= \frac{\alpha^2 u}{M_{ex}^2} (1 + v^2/4) \left[\frac{1}{(u-1)^2 + \epsilon} - \frac{(u-1)\, u\, v^2}{2\,[(u-1)^2 + \epsilon]^2}\right] \\ \langle\sigma v\rangle_{Taylor} &= \frac{\alpha^2 u}{M_{ex}^2} \left(1 + \frac{3}{2x}\right) \left[\frac{1}{(u-1)^2 + \epsilon} - \frac{(u-1)\, u\, 6/x}{2\,[(u-1)^2 + \epsilon]^2}\right]\end{aligned} \tag{43}$$

where $\langle v^2 \rangle = 6/x$ has been used.

Taylor expansion can be made after factoring off the pole factor $P(v^2) = \left[\left(1 - u/(1 - v^2/4)\right)^2 + \epsilon\right]^{-1}$. The resulting expansion is multiplied by the pole factor at $v = 0$

$$\begin{aligned}
(\sigma v)_0 &= \frac{\alpha^2 u}{M_{ex}^2(1 - v^2/4)}\, P(v^2 = 0) \\
&= \frac{\alpha^2 u}{M_{ex}^2}\left(1 + \frac{v^2}{4}\right) P(0) \\
\langle \sigma v \rangle_0 &= \frac{\alpha^2 u}{M_{ex}^2}\left(1 + \frac{3}{2x}\right) P(0) \\
&= \frac{\alpha^2 u}{M_{ex}^2}\left(1 + \frac{3}{2x}\right)\left[(1 - u)^2 + \epsilon\right]^{-1}
\end{aligned} \tag{44}$$

Another approximation to the cross-section is made by substituting $v^2 = 6/x$ directly in eq. (41)

$$\langle \sigma v \rangle_{subs} = (\sigma v)_{v^2 = 6/x} = \frac{\alpha^2 u/[1 - 3/(2x)]}{M_{ex}^2\left[\left(1 - u/[1 - 3/(2x)]\right)^2 + \epsilon\right]} \tag{45}$$

For small values of ϵ and $u < 1$, the cross-section can be expanded about the pole $v_p = 2\sqrt{1 - u}$. Changing the variable to $\upsilon' = v - v_p \implies dv = d\upsilon'$ and approximating $u/(1 - v^2/4) \approx 1 + \upsilon' \sqrt{1 - u}/u$, the cross-section eq. (42) is

$$\begin{aligned}
\langle \sigma v \rangle_{peak} &= \frac{x^{3/2}}{2\sqrt{\pi}} \int_{-v_p}^{\infty} d\upsilon'\, (\upsilon' + v_p)^2 \frac{\alpha^2}{M_{ex}^2}\, \frac{1 + \upsilon'\sqrt{1 - u}/u}{[\upsilon'\sqrt{1 - u}/u]^2 + \epsilon}\, e^{-x(\upsilon' + v_p)^2/4} \\
&\approx \frac{x^{3/2}}{2\sqrt{\pi}}\, v_p^2\, e^{-x v_p^2/4} \frac{\alpha^2}{M_{ex}^2} \int_{-v_0}^{v_0} d\upsilon'\, \frac{1}{[\upsilon'^2(1 - u)/u^2] + \epsilon} \qquad \text{for } \upsilon' \ll v_p \\
&= \frac{x^{3/2}}{2\sqrt{\pi}}\, v_p^2\, e^{-x v_p^2/4} \frac{\alpha^2}{M_{ex}^2} \frac{2u}{\sqrt{\epsilon\,(1 - u)}} \tan^{-1}\left(\sqrt{\frac{1 - u}{\epsilon\, u^2}}\, v_0\right)
\end{aligned} \tag{46}$$

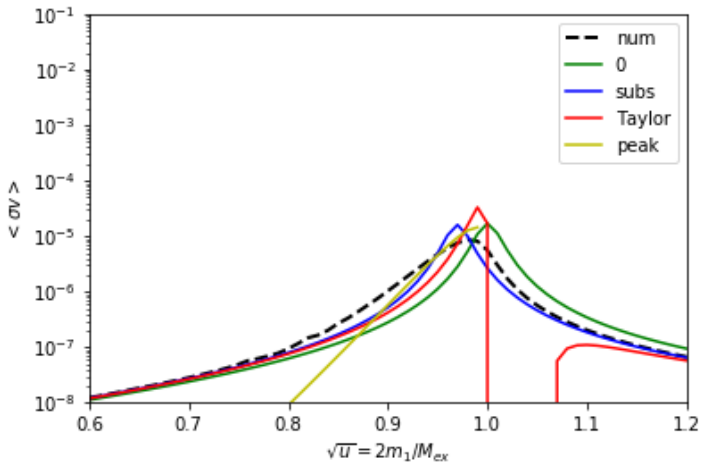
where v_0 is a small interval around the peak.

The comparison of the approximations with the numerical evaluation of the cross-section [eq. (42)] is shown in fig. 5. Fig. 5a shows the three approximations for $\epsilon = 7.5 \times 10^{-4}$ and $\alpha = 0.01$ which corresponds to a $Z-$boson exchange. Fig. 5b shows the approximations in comparison to the numerical integration for $\epsilon = 10^{-6}$.

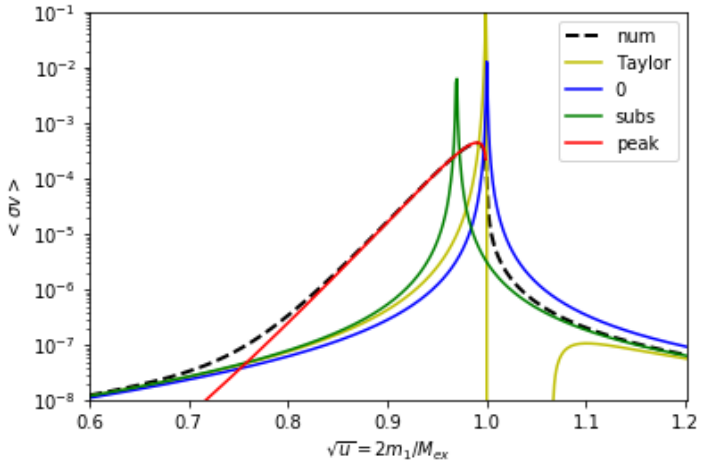
Annihilation integral and abundance

To calculate the abundance, the annihilation integral eq. (6) must be evaluated. The integral is written as

$$\begin{aligned}
J &= \int_0^{\infty} dv\, \frac{v^2\, (\sigma v)}{\sqrt{4\pi}} \int_{x_f}^{\infty} \frac{1}{\sqrt{x}} e^{-x v^2/4} \\
&= \int_0^{\infty} dv\, v\, (\sigma v)\, \mathrm{erfc}\left(\frac{v\sqrt{x_f}}{2}\right)
\end{aligned} \tag{47}$$



(a) $\epsilon = 7.5 \times 10^{-4}$



(b) $\epsilon = 10^{-8}$

Figure 5: Annihilation cross-section near a pole

For the cross-section eq. (41), the annihilation integral is

$$J_{num} = \int_0^\infty dv\ v\ \frac{\alpha^2 u/(1-v^2/4)}{M_{ex}^2\left[\left(1-u/(1-v^2/4)\right)^2+\epsilon\right]}\ \mathrm{erfc}\left(\frac{v\sqrt{x_f}}{2}\right) \tag{48}$$

The approximations of $\langle\sigma v\rangle_0$ [eq. (44)] and $\langle\sigma v\rangle_{subs}$ [eq. (45)] give

$$\begin{aligned} J_0 &= \int_{x_f}^\infty \frac{\langle\sigma v\rangle_0}{x^2}\,dx \\ &= \int_{x_f}^\infty \frac{\alpha^2 u}{M_{ex}^2}\left(1+\frac{3}{2x}\right)P(0)\ \frac{1}{x^2}\,dx \\ &= \frac{\alpha^2 u}{M_{ex}^2}\left(1+\frac{3}{4x_f}\right)\frac{1}{x_f}\ P(0) \end{aligned} \tag{49}$$

$$\begin{aligned} J_{subs} &= \int_{x_f}^\infty \frac{\langle\sigma v\rangle_{subs}}{x^2}\,dx \\ &\approx \langle\sigma v\rangle_{subs}\left(1-\frac{3}{4x_f}\right)\frac{1}{x_f} \\ &= \frac{\alpha^2 u/[1-3/(2x_f)]}{M_{ex}^2\left[\left(1-u/[1-3/(2x_f)]\right)^2+\epsilon\right]}\left(1-\frac{3}{4x_f}\right)\frac{1}{x_f} \end{aligned} \tag{50}$$

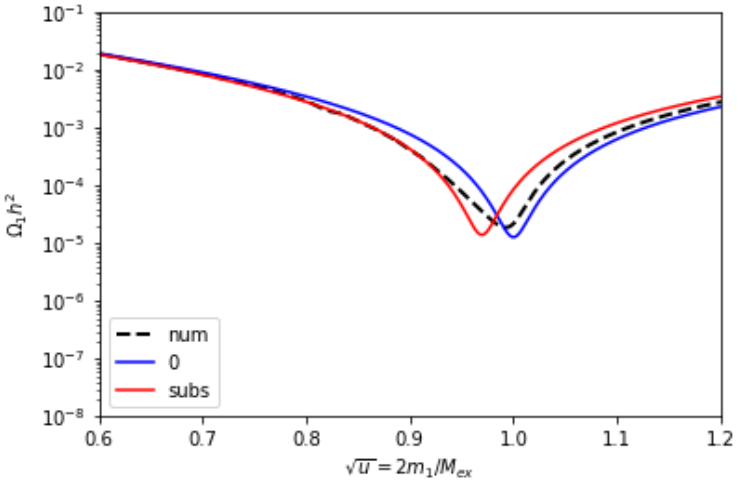
and $\langle\sigma v\rangle_{peak}$ gives

$$\begin{aligned} J_{peak} &= \frac{1}{2\sqrt{\pi}}\ v_p^2\ \frac{\alpha^2}{M_{ex}^2}\frac{2u}{\sqrt{\epsilon\,(1-u)}}\tan^{-1}\left(\sqrt{\frac{1-u}{\epsilon\,u^2}}v_0\right)\int_{x_f}^\infty e^{-xv_p^2/4}\frac{1}{x^2}\ x^{3/2}\,dx \\ &= \frac{1}{2\sqrt{\pi}}\ v_p^2\ \frac{\alpha^2}{M_{ex}^2}\frac{2u}{\sqrt{\epsilon\,(1-u)}}\tan^{-1}\left(\sqrt{\frac{1-u}{\epsilon\,u^2}}v_0\right)\frac{2\sqrt{\pi}}{v_p}\ \mathrm{erfc}(\sqrt{x_f}v_p/2) \\ &= \frac{4\alpha^2}{M_{ex}^2}\frac{u}{\sqrt{\epsilon}}\ \mathrm{erfc}\left(\sqrt{x_f(1-u)}\right)\tan^{-1}\left(\sqrt{\frac{1-u}{\epsilon\,u^2}}\ v_0\right) \end{aligned} \tag{51}$$

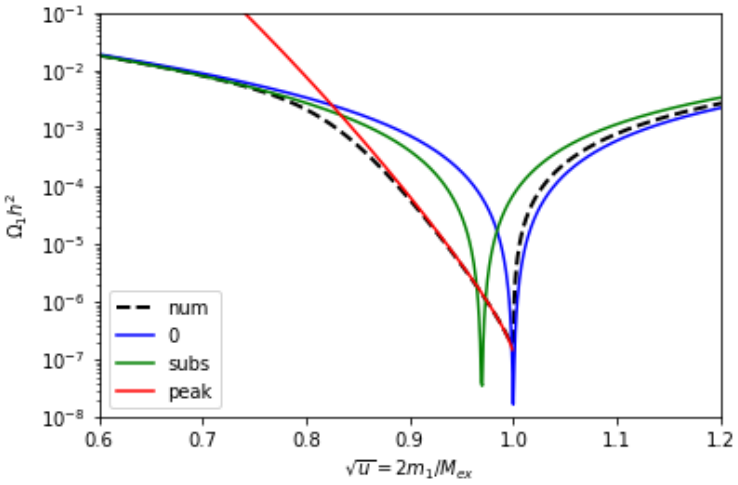
where $v_p = 2\sqrt{1-u}$ was used. The comparison of the numerical evaluation of J [eq. (48)] and the approximations [eq. (49), (50), (51)] are shown in fig. 6 for $\epsilon = 7.5\times10^{-4}$ and $\epsilon = 10^{-6}$. A link to the codes is given in appendix A.

Conclusion

With these results, the authors conclude that when annihilation happens near poles Taylor expansion cannot be used and special care must be taken. The effect is greater as the value of ϵ decreases. My results match the paper to a great extent.



(a) $\epsilon = 7.5 \times 10^{-4}$



(b) $\epsilon = 10^{-6}$

Figure 6: Relic abundance near a pole

5. Conclusion

Kim Griest and David Seckel (1991) found three exceptional cases where the standard method of calculating the relic abundance is inadequate. I have reviewed their paper and made an attempt to reproduce their results. On average my results are within 30% of theirs.

The first case, coannihilation, happens when other particles have mass nearly degenerate with the mass of the relic. It leads to annihilation of the heavier particles into the relic thereby affecting its abundance. This effect is particularly significant if the extra particle's mass is within 10% of the relic. Second, if the mass of the relic is 10–15% below the mass threshold, a significant number of annihilations may happen via forbidden channels. Lastly, if annihilation happens near a pole in the cross-section, the abundance curves are broader and more shallow than expected and Taylor expansion of the cross-section does not give good results. In all of these cases, a modified formula for the cross-section is necessary to accurately calculate the relic abundance.

A. Codes

Imported modules in Python

```
import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy.integrate import quad
from scipy.special import erfc
```

Figure 1: Comparison of Boltzmann equation with the standard method

```
MPl = 1.22e19
g_star = 106.75
m1 = 1000
g1 = 2
MPl_star = MPl/(1.66*np.sqrt(g_star))
H = 0.678
rho_c = 1.053e-5*H**2/(3e8)**2
omega_sm_arr = []
omega_rk_arr = []
xf_arr = []
error_arr = []
for xf in np.arange(20,30.5,0.5):
    xf_arr.append(xf)
    sigmav = np.exp(xf)*np.sqrt(xf)*np.sqrt(g_star)/(0.038*g1*MPl*m1)
# RK
    x0 = xf - 5
    n0 = g1*m1**3/(2*np.pi*x0)**(3/2)*np.exp(-x0)
    while x0<0.3e16:
        h = 0.0005*x0
        k1 = ( -3*n0/x0 - (MPl_star*sigmav/(m1**2))*x0*(n0**2-(g1*m1**3/(2*np.pi*x0)**(3/2)*np.exp(-x0))**2) )
        x01 = x0 + h/2
        n01 = n0 + h*k1/2
        k2 = ( -3*n01/x01 - (MPl_star*sigmav/(m1**2))*x01*(n01**2-(g1*m1**3/(2*np.pi*x01)**(3/2)*np.exp(-x01))**2) )
        x02 = x0 + h/2
        n02 = n0 + h*k2/2
        k3 = ( -3*n02/x02 - (MPl_star*sigmav/(m1**2))*x02*(n02**2-(g1*m1**3/(2*np.pi*x02)**(3/2)*np.exp(-x02))**2) )
```

```
        x03 = x0 + h/2
        n03 = n0 + h*k1
        k4 = ( -3*n03/x03 - (MPl_star*sigmav/(ml**2))*x03*(n03**2-(g1*ml**3/(2*np.pi*x03)**(3/2)*np.exp(-x03))**2) )
        n0 = n0 + h*(1/6)*(k1+2*k2+2*k3+k4)
        x0 = x0 + h
    omega_rk = ml*n0*H**2/rho_c**2
    omega_rk_arr.append(omega_rk)
# SM
    omega_sm = 1.07e9*xf/(g_star**0.5*MPl*sigmav)
    omega_sm_arr.append(omega_sm)
# error
    error = np.abs((omega_sm - omega_rk)/omega_rk)*100
    error_arr.append(error)
plt.plot(xf_arr, omega_sm_arr, 'c', label='Standard method')
plt.plot(xf_arr, omega_rk_arr, 'xk', label='Numerical method')
plt.legend()
plt.yscale("log")
plt.xlabel(r"$x_f$")
plt.ylabel(r'$\Omega h^2$')
```

Codes for the three exceptional cases

Link to Google Drive folder with codes

Bibliography

[1] Griest, Kim & Seckel, David. (1991). Three exceptions in the calculation of relic abundances. Physical review D: Particles and fields. 43. 3191-3203. 10.1103/PhysRevD.43.3191.

1 Modules

```
import numpy as np
import matplotlib.pyplot as plt
import scipy
from scipy.integrate import quad
from scipy.special import erfc
```

2 Figure 1

```
MPl = 1.22e19
g_star = 106.75
m1 = 1000
g1 = 2
MPl_star = MPl/(1.66*np.sqrt(g_star))
H = 0.678
rho_c = 1.053e-5*H**2/(3e8)**2
omega_sm_arr = []
omega_rk_arr = []
xf_arr = []
error_arr = []
for xf in np.arange(20,30.5,0.5):
    xf_arr.append(xf)
    sigmav = np.exp(xf)*np.sqrt(xf)*np.sqrt(g_star)/(0.038*g1*MPl*m1)
# RK
    x0 = xf - 5
    n0 = g1*m1**3/(2*np.pi*x0)**(3/2)*np.exp(-x0)
    while x0<0.3e16:
        h = 0.0005*x0
        k1 = ( -3*n0/x0 - (MPl_star*sigmav/(m1**2))*x0*(n0**2-(g1*m1**3/(2*np.pi*x0)**(3/2)*
            np.exp(-x0))**2) )
        x01 = x0 + h/2
        n01 = n0 + h*k1/2
        k2 = ( -3*n01/x01 - (MPl_star*sigmav/(m1**2))*x01*(n01**2-(g1*m1**3/(2*np.pi*x01)
            **(3/2)*np.exp(-x01))**2) )
        x02 = x0 + h/2
        n02 = n0 + h*k2/2
        k3 = ( -3*n02/x02 - (MPl_star*sigmav/(m1**2))*x02*(n02**2-(g1*m1**3/(2*np.pi*x02)
            **(3/2)*np.exp(-x02))**2) )
        x03 = x0 + h/2
        n03 = n0 + h*k1
        k4 = ( -3*n03/x03 - (MPl_star*sigmav/(m1**2))*x03*(n03**2-(g1*m1**3/(2*np.pi*x03)
            **(3/2)*np.exp(-x03))**2) )
        n0 = n0 + h*(1/6)*(k1+2*k2+2*k3+k4)
        x0 = x0 + h
    omega_rk = m1*n0*H**2/rho_c**2
    omega_rk_arr.append(omega_rk)
# SM
    omega_sm = 1.07e9*xf/(g_star**0.5*MPl*sigmav)
    omega_sm_arr.append(omega_sm)
# error
    error = np.abs((omega_sm - omega_rk)/omega_rk)*100
    error_arr.append(error)
plt.plot(xf_arr, omega_sm_arr, 'c', label='Standard method')
plt.plot(xf_arr, omega_rk_arr, 'xk', label='Numerical method')
plt.legend()
plt.yscale("log")
plt.xlabel(r"$x_f$")
plt.ylabel(r'$\Omega_h^2$')
```

3 Figure 2

3.1 2a

```
A = 20
g_2_g_1 = 3
xf_old1 = 20
xf_old2 = 25
xf_old3 = 30
MPl = 1.22e19
g_star = 106.75
m1 = 10000
sigmav_eff1 = np.exp(xf_old1)*np.sqrt(xf_old1)*np.sqrt(g_star)/(0.38*g1*MPl*m1)
sigmav_eff2 = np.exp(xf_old2)*np.sqrt(xf_old2)*np.sqrt(g_star)/(0.38*g1*MPl*m1)
sigmav_eff3 = np.exp(xf_old3)*np.sqrt(xf_old3)*np.sqrt(g_star)/(0.38*g1*MPl*m1)
g1 = 2
oo = np.inf
R_arr1 = []
R_arr2 = []
R_arr3 = []
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff1/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old1
    f3 = lambda x : (1/x**2)*(1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)**2/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
    f4 = lambda x : (6/x**3)*(1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)**2/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+A*(1+delta)*np
            .exp(-x*delta)*g_2_g_1)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<1.5e6:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old1/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr1.append(R)
            break
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff2/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old2
    f3 = lambda x : (1/x**2)*(1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)**2/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
    f4 = lambda x : (6/x**3)*(1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)**2/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+A*(1+delta)*np
            .exp(-x*delta)*g_2_g_1)**2
        a11 = 10
        b11 = 10
```

```
        a = 10
        b = 10
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old2/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr2.append(R)
            break
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff3/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old3
    f3 = lambda x : (1/x**2)*(1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)**2/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
    f4 = lambda x : (6/x**3)*(1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)**2/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+A*(1+delta)*np
            .exp(-x*delta)*g_2_g_1)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<5e10:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old3/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr3.append(R)
            break
MPl_star = MPl/(1.66*np.sqrt(g_star))
g2_g1 = 3
A = 20
xf_old = 25
sigmav = 4e-11
H = 0.678
rho_c = 1.053e-5*H**2
val = 0.038*g1*MPl*m1*sigmav/np.sqrt(g_star)
delta_rk_arr = []
R_arr = []
omega_old_arr = []
omega_new_arr = []
for delta in np.arange(0.02,0.25,0.02):
    print('====================',delta,'====================')
    delta_rk_arr.append(delta)
    x_old = 25 - 2
    n_old = g1*m1**3/(2*np.pi*x_old)**(3/2)*np.exp(-x_old)
    while x_old<0.3e16:
        h = 0.0005*x_old
        k1 = ( -3*n_old/x_old - (MPl_star*sigmav/(m1**2))*x_old*(n_old**2-(g1*m1**3/(2*np.pi
            *x_old)**(3/2)*np.exp(-x_old))**2) )
        x_old1 = x_old + h/2
        n_old1 = n_old + h*k1/2
```

```
        k2 = ( -3*n_old1/x_old1 - (MPl_star*sigmav/(m1**2))*x_old1*(n_old1**2-(g1*m1**3/(2*
            np.pi*x_old1)**(3/2)*np.exp(-x_old1))**2) )
        x_old2 = x_old + h/2
        n_old2 = n_old + h*k2/2
        k3 = ( -3*n_old2/x_old2 - (MPl_star*sigmav/(m1**2))*x_old2*(n_old2**2-(g1*m1**3/(2*
            np.pi*x_old2)**(3/2)*np.exp(-x_old2))**2) )
        x_old3 = x_old + h/2
        n_old3 = n_old + h*k1
        k4 = ( -3*n_old3/x_old3 - (MPl_star*sigmav/(m1**2))*x_old3*(n_old3**2-(g1*m1**3/(2*
            np.pi*x_old3)**(3/2)*np.exp(-x_old3))**2) )
        n_old = n_old + h*(1/6)*(k1+2*k2+2*k3+k4)
        x_old = x_old + h
    omega_old = m1*n_old*H**2/rho_c**2
    omega_old_arr.append(omega_old)
    for y in np.arange(5,40,0.01):
        f = y**0.5*np.exp(y)*(1+(1+delta)**(3/2)*np.exp(-y*delta)*g2_g1)/(1+A*(1+delta)
            **(3/2)*np.exp(-y*delta)*g2_g1)**2
        if np.abs(f-val)<2.5e8:
            x_new = y
            n_new = g1*m1**3/(2*np.pi*x_new)**(3/2)*np.exp(-x_new)
            while x_new < x_old:
                h = 0.0005*x_new
                k1 = ( -3*n_new/x_new - (MPl_star*sigmav*((1+A*(1+delta)**(3/2)*g2_g1*np.exp
                    (-x_new*delta))/(1+(1+delta)**(3/2)*g2_g1*np.exp(-x_new*delta)))**2/(m1
                    **2))*x_new*(n_new**2-(g1*m1**3/(2*np.pi*x_new)**(3/2)*np.exp(-x_new))
                    **2) )
                x_new1 = x_new + h/2
                n_new1 = n_new + h*k1/2
                k2 = ( -3*n_new1/x_new1 - (MPl_star*sigmav*((1+A*(1+delta)**(3/2)*g2_g1*np.
                    exp(-x_new1*delta))/(1+(1+delta)**(3/2)*g2_g1*np.exp(-x_new1*delta)))
                    **2/(m1**2))*x_new1*(n_new1**2-(g1*m1**3/(2*np.pi*x_new1)**(3/2)*np.exp(-
                    x_new1))**2) )
                x_new2 = x_new + h/2
                n_new2 = n_new + h*k2/2
                k3 = ( -3*n_new2/x_new2 - (MPl_star*sigmav*((1+A*(1+delta)**(3/2)*g2_g1*np.
                    exp(-x_new2*delta))/(1+(1+delta)**(3/2)*g2_g1*np.exp(-x_new2*delta)))
                    **2/(m1**2))*x_new2*(n_new2**2-(g1*m1**3/(2*np.pi*x_new2)**(3/2)*np.exp(-
                    x_new2))**2) )
                x_new3 = x_new + h/2
                n_new3 = n_new + h*k3/2
                k4 = ( -3*n_new3/x_new3 - (MPl_star*sigmav*((1+A*(1+delta)**(3/2)*g2_g1*np.
                    exp(-x_new3*delta))/(1+(1+delta)**(3/2)*g2_g1*np.exp(-x_new3*delta)))
                    **2/(m1**2))*x_new3*(n_new3**2-(g1*m1**3/(2*np.pi*x_new3)**(3/2)*np.exp(-
                    x_new3))**2) )
                n_new = n_new + h*(1/6)*(k1+2*k2+2*k3+k4)
                x_new = x_new + h
            omega_new = m1*n_new*H**2/rho_c**2
            omega_new_arr.append(omega_new)
            break
    R = omega_old/omega_new
    R_arr.append(R)
plt.plot(delta_rk_arr,R_arr,'xk',label='Numerical Integration')
plt.plot(delta_arr,R_arr1,'b',label='xf_old = 20')
plt.plot(delta_arr,R_arr2,'r',label='xf_old = 25')
plt.plot(delta_arr,R_arr3,'g',label='xf_old = 30')
plt.ylim(1,300)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$\Delta = (m_2-m_1)/m_1$")
plt.ylabel(r'$\Omega_{old}/\Omega_{new}$')
```

3.2 2b

```
A1 = 10
A2 = 20
A3 = 40
A4 = 80
g_2_g_1 = 3
xf_old = 25
MPl = 1.22e19
g_star = 106.75
m1 = 10000
sigmav_eff2 = np.exp(xf_old2)*np.sqrt(xf_old2)*np.sqrt(g_star)/(0.38*g1*MPl*m1)
g1 = 2
oo = np.inf
R_arr1 = []
R_arr2 = []
R_arr3 = []
R_arr4 = []
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff2/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old
    f3 = lambda x : (1/x**2)*((1+A1*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    f4 = lambda x : (6/x**3)*((1+A1*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+A1*(1+delta)
            **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr1.append(R)
            break
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff2/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old
    f3 = lambda x : (1/x**2)*((1+A2*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    f4 = lambda x : (6/x**3)*((1+A2*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+A2*(1+delta)
            **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
```

```
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr2.append(R)
            break
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff2/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old
    f3 = lambda x : (1/x**2)*((1+A3*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    f4 = lambda x : (6/x**3)*((1+A3*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+A3*(1+delta)
            **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr3.append(R)
            break
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff2/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old
    f3 = lambda x : (1/x**2)*((1+A4*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    f4 = lambda x : (6/x**3)*((1+A4*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_1))**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_1)/(1+A4*(1+delta)
            **(3/2)*np.exp(-x*delta)*g_2_g_1)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr4.append(R)
            break
```

```
plt.plot(delta_arr,R_arr1,'b',label='A = 10')
plt.plot(delta_arr,R_arr2,'r',label='A = 20')
plt.plot(delta_arr,R_arr3,'y',label='A = 40')
plt.plot(delta_arr,R_arr4,'g',label='A = 80')
plt.ylim(1,300)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$\Delta = (m_2-m_1)/m_1$")
plt.ylabel(r'$\Omega_{old}/\Omega_{new}$')
```

3.3 2c

```
A = 20
g_2_g_11 = 2
g_2_g_12 = 6
g_2_g_13 = 18
xf_old = 25
MPl = 1.22e19
g_star = 106.75
m1 = 10000
sigmav_eff = np.exp(xf_old)*np.sqrt(xf_old)*np.sqrt(g_star)/(0.38*g1*MPl*m1)
g1 = 2
oo = np.inf
R_arr1 = []
R_arr2 = []
R_arr3 = []
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old
    f3 = lambda x : (1/x**2)*((1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_11)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_11))**2
    f4 = lambda x : (6/x**3)*((1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_11)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_11))**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_11)/(1+A*(1+delta)
            **(3/2)*np.exp(-x*delta)*g_2_g_11)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr1.append(R)
            break
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old
    f3 = lambda x : (1/x**2)*((1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_12)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_12))**2
```

```
    f4 = lambda x : (6/x**3)*((1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_12)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_12))**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_12)/(1+A*(1+delta)
            **(3/2)*np.exp(-x*delta)*g_2_g_12)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr2.append(R)
            break
delta_arr = []
for delta in np.arange(0,0.41,0.01):
    delta_arr.append(delta)
    val = 0.038*g1*MPl*m1*sigmav_eff/np.sqrt(g_star)
    f1 = 1
    f2 = 6/xf_old
    f3 = lambda x : (1/x**2)*((1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_13)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_13))**2
    f4 = lambda x : (6/x**3)*((1+A*(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_13)/(1+(1+delta)
        **(3/2)*np.exp(-x*delta)*g_2_g_13))**2
    for x in np.arange(5,40,0.01):
        f = x**0.5*np.exp(x)*(1+(1+delta)**(3/2)*np.exp(-x*delta)*g_2_g_13)/(1+A*(1+delta)
            **(3/2)*np.exp(-x*delta)*g_2_g_13)**2
        a11 = 10
        b11 = 10
        a = 10
        b = 10
        if np.abs(f-val)<2.5e8:
            xf_new = x
            Ia_new = quad( f3, xf_new,oo )
            Ib_new = quad( f4, xf_new,oo )
            omega_old = xf_old/(a*f1+b*f2)
            omega_new = 1/(a11*Ia_new[0] + b11*Ib_new[0])
            R = omega_old / omega_new
            R_arr3.append(R)
            break
plt.plot(delta_arr,R_arr1,'g',label=r'$g_2/g_1 = 2$')
plt.plot(delta_arr,R_arr2,'r',label=r'$g_2/g_1 = 6$')
plt.plot(delta_arr,R_arr3,'b',label=r'$g_2/g_1 = 18$')
plt.ylim(1,300)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$\Delta = (m_2-m_1)/m_1$")
plt.ylabel(r'$\Omega_{old}/\Omega_{new}$')
```

4 Figure 3

4.1 3a

```
x = 25
zA_arr = []
```

```
zT_arr = []
zF_arr = []
A_arr = []
T_arr = []
F_arr = []
intA_arr = []
intF_arr = []
for z in np.arange(0.8,0.999,0.001):
    zA_arr.append(z)
    A = np.sqrt(1-z**2)*( 1 + (3*z**2)/(4*x*(1-z**2)) )
    A_arr.append(A)
    f = lambda t : 2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(1-z**2)*x*t/z**2)*np.exp(-t)
    integral = quad( f, 0,oo )
    intA_arr.append(integral[0])
for z in np.arange(0.8,1.201,0.001):
    zT_arr.append(z)
    T = 2*z/np.sqrt(np.pi*x)*( 1 - x*(z-1) )
    T_arr.append(T)
for z in np.arange(1.001,1.201,0.001):
    zF_arr.append(z)
    F = np.sqrt(z**2-1)*np.exp(-(z**2-1)*x/z**2)*( 1 + 3*z**2/(4*x*(z**2-1)) )
    F_arr.append(F)
    f = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(z**2-1)*t*x/z
        **2)*np.exp(-t)
    integral = quad( f, 0,oo )
    intF_arr.append(integral[0])
plt.plot(zA_arr, intA_arr, '--k', label='Numerical Integration',linewidth=2)
plt.plot(zF_arr, intF_arr, '--k',linewidth=2)
plt.plot(zA_arr, A_arr, 'r', label='Allowed')
plt.plot(zT_arr, T_arr, 'g', label='Threshold')
plt.plot(zF_arr, F_arr, 'c', label='Forbidden')
plt.ylim(0.001,3)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$z = m_2/m_1$")
plt.ylabel(r'$< \sigma v >/a$')
```

4.2 3b

```
x = 25
zA_arr = []
zT_arr = []
zF_arr = []
A_arr = []
T_arr = []
F_arr = []
intA_arr = []
intF_arr = []
for z in np.arange(0.8,0.999,0.001):
    zA_arr.append(z)
    A = np.sqrt(1-z**2)*( 1 + (5*z**2)/(4*x*(1-z**2)) )
    A_arr.append(A)
    f = lambda t : 2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(1-z**2)*x*t/z**2)*(2/3)*t*np.exp(-t)
    integral = quad( f, 0,oo )
    intA_arr.append(integral[0])
for z in np.arange(0.8,1.201,0.001):
    zT_arr.append(z)
    T = 2*z/np.sqrt(np.pi*x)*(4/3)*( 1 - x*(z-1)/2 )
    T_arr.append(T)
for z in np.arange(1.001,1.201,0.001):
```

```
    zF_arr.append(z)
    F = np.sqrt(z**2-1)*np.exp(-(z**2-1)*x/z**2)*(2*x/3)*((z**2-1)/z**2)*( 1 + 9*z**2/(4*x*(
        z**2-1)) + 45*z**4/(32*x**2*(z**2-1)**2) )
    F_arr.append(F)
    f = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(z**2-1)*t*x/z
        **2)*np.exp(-t)*( 2*t/3 + 2*(z**2-1)*x/(3*z**2) )
    integral = quad( f, 0,oo )
    intF_arr.append(integral[0])
plt.plot(zA_arr, intA_arr, '--k', label='Numerical Integration',linewidth=2)
plt.plot(zF_arr, intF_arr, '--k',linewidth=2)
plt.plot(zA_arr, A_arr, 'r', label='Allowed')
plt.plot(zT_arr, T_arr, 'g', label='Threshold')
plt.plot(zF_arr, F_arr, 'c', label='Forbidden')
plt.ylim(0.001,3)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$z = m_2/m_1$")
plt.ylabel(r'$< \sigma_v >x/6b$')
```

5 Figure 4

5.1 4a

```
g1 = 2
g_star = 106.75
MPl = 1.22e19
m1 = 1000
A1 = 20
A2 = 100
A3 = 500
xf_old = 25
val = 0.038*g1*MPl*m1/np.sqrt(g_star)
RA_arr = []
RT_arr = []
RF_arr = []
z_arr1 = []
z_arr2 = []
z_arr3 = []
num1_arr1 = []
num2_arr1 = []
num3_arr1 = []
znum_arr11 = []
znum_arr21 = []
znum_arr31 = []
for z in np.arange(0.8,1.0,0.001):
    z_arr1.append(z)
    for x in np.arange(10,50,0.01):
        sigmav = np.sqrt(1-z**2)*(1+3*z**2/(4*x*(1-z**2)))
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<5e17:
            xf_new = x
            integrate = lambda x : sigmav/x**2
            Ia = quad( integrate, xf_new,oo )
            RA = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            RA_arr.append(RA)
            break
        elif np.abs(f+val)<5e17:
            xf_new = x
            integrate = lambda x : sigmav/x**2
```

```
            Ia = quad( integrate, xf_new,oo )
            RA = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            RA_arr.append(RA)
            break
for z in np.arange(0.8,1.201,0.001):
    z_arr2.append(z)
    for x in np.arange(10,50,0.01):
        sigmav = (2*z/np.sqrt(np.pi*x))*(1-x*(z-1))
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<7.5e17:
            xf_new = x
            integrate = lambda x : sigmav/x**2
            Ia = quad( integrate, xf_new,oo )
            RT = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            RT_arr.append(RT)
            break
        elif np.abs(f+val)<7.5e17:
            xf_new = x
            integrate = lambda x : sigmav/x**2
            Ia = quad( integrate, xf_new,oo )
            RT = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            RT_arr.append(RT)
            break
for z in np.arange(1.001,1.201,0.001):
    z_arr3.append(z)
    for x in np.arange(10,50,0.01):
        sigmav = np.sqrt(z**2-1)*np.exp(-(z**2-1)*x/z**2)*(1+3*z**2/(4*x*(z**2-1)))
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<6e17:
            xf_new = x
            integrate = lambda x : sigmav/x**2
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            RF_arr.append(RF)
            break
        elif np.abs(f+val)<6e17:
            xf_new = x
            integrate = lambda x : sigmav/x**2
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            RF_arr.append(RF)
            break
#=============================================
#=============================================
for z in np.arange(0.8,1.0,0.01):
    znum_arr11.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : 2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(1-z**2)*x*t/z**2)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/x**2
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<5e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RA = (1+A1*xf_new*Ia[0])*xf_old/xf_new
            num1_arr1.append(RA)
            break
        elif np.abs(f+val)<5e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
```

```
            RA = (1+A1*xf_new*Ia[0])*xf_old/xf_new
            num1_arr1.append(RA)
            break
for z in np.arange(1.0,1.201,0.01):
    znum_arr11.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(z**2-1)
            *t*x/z**2)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/x**2
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<6e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A1*xf_new*Ia[0])*xf_old/xf_new
            num1_arr1.append(RF)
            break
        elif np.abs(f+val)<6e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A1*xf_new*Ia[0])*xf_old/xf_new
            num1_arr1.append(RF)
            break
for z in np.arange(0.8,1.0,0.01):
    znum_arr21.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : 2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(1-z**2)*x*t/z**2)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/x**2
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<5e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RA = (1+A2*xf_new*Ia[0])*xf_old/xf_new
            num2_arr1.append(RA)
            break
        elif np.abs(f+val)<5e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RA = (1+A2*xf_new*Ia[0])*xf_old/xf_new
            num2_arr1.append(RA)
            break
for z in np.arange(1.0,1.201,0.01):
    znum_arr21.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(z**2-1)
            *t*x/z**2)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/x**2
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<6e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A2*xf_new*Ia[0])*xf_old/xf_new
            num2_arr1.append(RF)
            break
        elif np.abs(f+val)<6e18:
            xf_new = x
```

```
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A2*xf_new*Ia[0])*xf_old/xf_new
            num2_arr1.append(RF)
            break
for z in np.arange(0.8,1.0,0.01):
    znum_arr31.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : 2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(1-z**2)*x*t/z**2)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/x**2
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<5e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RA = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            num3_arr1.append(RA)
            break
        elif np.abs(f+val)<5e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RA = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            num3_arr1.append(RA)
            break
for z in np.arange(1.0,1.201,0.01):
    znum_arr31.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi*x)*np.sqrt(t**2+(z**2-1)
            *t*x/z**2)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/x**2
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<6e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            num3_arr1.append(RF)
            break
        elif np.abs(f+val)<6e18:
            xf_new = x
            Ia = quad( integrate, xf_new,oo )
            RF = (1+A3*xf_new*Ia[0])*xf_old/xf_new
            num3_arr1.append(RF)
            break
plt.plot(z_arr1, RA_arr, '--k', linewidth=2)
plt.plot(z_arr2, RT_arr, '--k', linewidth=2)
plt.plot(z_arr3, RF_arr, '--k', label='Analytic Approximation', linewidth=2)
plt.plot(znum_arr11, num1_arr1, 'g', label='A=20')
plt.plot(znum_arr21, num2_arr1, 'c', label='A=100')
plt.plot(znum_arr31, num3_arr1, 'r', label='A=500')
plt.ylim(0.7,1000)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$z = m_2/m_1$")
plt.ylabel(r'$\Omega_{old}/\Omega_{new}$')
```

5.2 4b

```
g1 = 2
```

```
g_star = 106.75
MPl = 1.22e19
m1 = 1000
A1 = 20
A2 = 100
A3 = 500
xf_old = 25
val = 0.038*g1*MPl*m1/np.sqrt(g_star)
RA_arr = []
RT_arr = []
RF_arr = []
z_arr1 = []
z_arr2 = []
z_arr3 = []
num1_arr1 = []
num2_arr1 = []
num3_arr1 = []
znum_arr11 = []
znum_arr21 = []
znum_arr31 = []
for z in np.arange(0.8,1.0,0.001):
    z_arr1.append(z)
    for x in np.arange(10,50,0.01):
        sigmav = np.sqrt(1-z**2)*(6/x)*(1+5*z**2/(4*x*(1-z**2)))
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<5e17:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            RA_arr.append(RA)
            break
        elif np.abs(f+val)<5e17:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            RA_arr.append(RA)
            break
for z in np.arange(0.8,1.201,0.004):
    z_arr2.append(z)
    for x in np.arange(10,50,0.01):
        sigmav = (2*z/np.sqrt(np.pi*x))*(8/x)*(1-x*(z-1)/2)
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<7.5e17:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RT = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            RT_arr.append(RT)
            break
        elif np.abs(f+val)<7.5e17:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RT = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            RT_arr.append(RT)
            break
for z in np.arange(1.001,1.201,0.001):
    z_arr3.append(z)
    for x in np.arange(10,50,0.01):
        sigmav = np.sqrt(z**2-1)*np.exp(-(z**2-1)*x/z**2)*(4*(z**2-1)/z**2)*(1+9*z**2/(4*x*(
            z**2-1))+45*z**4/(32*(z**2-1)**2*x**2))
```

```
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<6e17:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            RF_arr.append(RF)
            break
        elif np.abs(f+val)<6e17:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            RF_arr.append(RF)
            break
#=============================================================
#=============================================================
for z in np.arange(0.8,1.0,0.01):
    znum_arr11.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : 2*z/np.sqrt(np.pi)*np.sqrt(t/x+(1-z**2)/z**2)*np.sqrt(t)*(4*t/x)*np
            .exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A1*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num1_arr1.append(RA)
            break
        elif np.abs(f+val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A1*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num1_arr1.append(RA)
            break
for z in np.arange(1.0,1.21,0.01):
    znum_arr11.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi)*np.sqrt(t/x+(z**2-1)/z
            **2)*np.sqrt(t)*(4*(z**2-1)/z**2+4*t/x)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A1*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num1_arr1.append(RF)
            break
        elif np.abs(f+val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A1*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num1_arr1.append(RF)
            break
for z in np.arange(0.8,1.0,0.01):
    znum_arr21.append(z)
    for x in np.arange(10,50,0.1):
```

```
        svf = lambda t : 2*z/np.sqrt(np.pi)*np.sqrt(t/x+(1-z**2)/z**2)*np.sqrt(t)*(4*t/x)*np
            .exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A2*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num2_arr1.append(RA)
            break
        elif np.abs(f+val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A2*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num2_arr1.append(RA)
            break
for z in np.arange(1.0,1.21,0.01):
    znum_arr21.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi)*np.sqrt(t/x+(z**2-1)/z
            **2)*np.sqrt(t)*(4*(z**2-1)/z**2+4*t/x)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A2*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num2_arr1.append(RF)
            break
        elif np.abs(f+val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A2*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num2_arr1.append(RF)
            break
for z in np.arange(0.8,1.0,0.01):
    znum_arr31.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : 2*z/np.sqrt(np.pi)*np.sqrt(t/x+(1-z**2)/z**2)*np.sqrt(t)*(4*t/x)*np
            .exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num3_arr1.append(RA)
            break
        elif np.abs(f+val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RA = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num3_arr1.append(RA)
            break
for z in np.arange(1.0,1.21,0.01):
```

```
    znum_arr31.append(z)
    for x in np.arange(10,50,0.1):
        svf = lambda t : np.exp(-(z**2-1)*x/z**2)*2*z/np.sqrt(np.pi)*np.sqrt(t/x+(z**2-1)/z
            **2)*np.sqrt(t)*(4*(z**2-1)/z**2+4*t/x)*np.exp(-t)
        svint = quad( svf, 0,oo )
        sigmav = svint[0]
        integrate = lambda x : sigmav/(6*x**2)
        f = np.exp(x)*np.sqrt(x)/sigmav
        if np.abs(f-val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num3_arr1.append(RF)
            break
        elif np.abs(f+val)<2e19:
            xf_new = x
            Ib = quad( integrate, xf_new,oo )
            RF = (1+A3*2*xf_new**2*Ib[0])*xf_old**2/xf_new**2
            num3_arr1.append(RF)
            break
plt.plot(z_arr1, RA_arr, '--k', linewidth=2)
plt.plot(z_arr2, RT_arr, '--k', linewidth=2)
plt.plot(z_arr3, RF_arr, '--k', label='Analytic Approximation', linewidth=2)
plt.plot(znum_arr11, num1_arr1, 'g', label='A=20')
plt.plot(znum_arr21, num2_arr1, 'c', label='A=100')
plt.plot(znum_arr31, num3_arr1, 'r', label='A=500')
plt.ylim(0.52,1000)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$z = m_2/m_1$")
plt.ylabel(r'$\Omega_{old}/\Omega_{new}$')
```

6 Figure 5

6.1 5a

```
epsilon1 = 7.5e-4
epsilon2 = 1e-8
M_ex = 91
x = 25
alpha = 0.01
sqrt_u_arr = []
sigmav_0_arr = []
sigmav_subs_arr = []
sigmav_taylor_arr = []
sigmav_peak_arr = []
sigmav_num_arr = []
for sqrt_u in np.arange(0.6,1.21,0.01):
    sqrt_u_arr.append(sqrt_u)
#=========
    sigmav_0 = (alpha**2*sqrt_u**2/M_ex**2)*(1+3/(2*x))*(1/( (1-sqrt_u**2)**2+epsilon1 ))
    sigmav_0_arr.append(sigmav_0)
#=========
    sigmav_subs = ( alpha**2*sqrt_u**2/(1-3/(2*x)) )/( M_ex**2*((1-sqrt_u**2/(1-3/(2*x)))
        **2+epsilon1) )
    sigmav_subs_arr.append(sigmav_subs)
#=========
    sigmav_taylor = (alpha**2*sqrt_u**2/M_ex**2)*(1+3/(2*x))*( 1/((sqrt_u**2-1)**2+epsilon1)
        - (sqrt_u**2-1)*sqrt_u**2*(6/x)/(2*((sqrt_u**2-1)**2+epsilon1)**2) )
```

```
    sigmav_taylor_arr.append(sigmav_taylor)
#=========
    sigmav_peak = (x**(3/2)/(2*np.sqrt(np.pi)))*4*(1-sqrt_u**2)*np.exp(-x*(1-sqrt_u**2))*(
        alpha**2/M_ex**2)*(2*sqrt_u**2/np.sqrt(epsilon1*(1-sqrt_u**2)))*np.arctan(np.sqrt((1-
        sqrt_u**2)/(epsilon1*sqrt_u**4)))
    sigmav_peak_arr.append(sigmav_peak)
#=========
    integrand = lambda v : v**2*np.exp(-x*v**2/4)*( alpha**2*sqrt_u**2/(1-v**2/4) )/( M_ex
        **2*((1-sqrt_u**2/(1-v**2/4))**2+epsilon1) )
    integral = quad( integrand, 0,oo )
    sigmav_num = (x**(3/2)/(2*np.sqrt(np.pi)))*integral[0]
    sigmav_num_arr.append(sigmav_num)
plt.plot(sqrt_u_arr, sigmav_num_arr, '--k', label=r'num',linewidth=2)
plt.plot(sqrt_u_arr, sigmav_0_arr, 'g', label=r'0')
plt.plot(sqrt_u_arr, sigmav_subs_arr, 'b', label=r'subs')
plt.plot(sqrt_u_arr, sigmav_taylor_arr, 'r', label=r'Taylor')
plt.plot(sqrt_u_arr, sigmav_peak_arr, 'y', label='peak')
plt.ylim(1e-8,0.1)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$\sqrt{u} = 2m_1/M_{ex}$")
plt.ylabel(r'$< \sigma_v >$')
```

6.2 5b

```
epsilon = 1e-6
M_ex = 91
x = 25
alpha = 0.01
sqrt_u_arr = []
sigmav_0_arr = []
sigmav_subs_arr = []
sigmav_num_arr = []
sigmav_peak_arr = []
sigmav_taylor_arr = []
for sqrt_u in np.arange(0.6,1.201,0.001):
    sqrt_u_arr.append(sqrt_u)
#=========
    sigmav_0 = (alpha**2*sqrt_u**2/M_ex**2)*(1+3/(2*x))*(1/( (1-sqrt_u**2)**2+epsilon ))
    sigmav_0_arr.append(sigmav_0)
#=========
    sigmav_subs = ( alpha**2*sqrt_u**2/(1-3/(2*x)) )/( M_ex**2*((1-sqrt_u**2/(1-3/(2*x)))
        **2+epsilon) )
    sigmav_subs_arr.append(sigmav_subs)
#=========
    integrand = lambda v : (x**(3/2)/(2*np.sqrt(np.pi)))*v**2*np.exp(-x*v**2/4)*(sqrt_u
        **2/(1-v**2/4))/(((1-sqrt_u**2/(1-v**2/4))**2+epsilon))
    integral = quad( integrand, 0,oo )
    sigmav_num = integral[0]*(alpha**2/M_ex**2)
    sigmav_num_arr.append(sigmav_num)
#=========
    sigmav_peak = (x**(3/2)/(2*np.sqrt(np.pi)))*4*(1-sqrt_u**2)*np.exp(-x*(1-sqrt_u**2))*(
        alpha**2/M_ex**2)*(2*sqrt_u**2/np.sqrt(epsilon*(1-sqrt_u**2)))*np.arctan(np.sqrt((1-
        sqrt_u**2)/(epsilon*sqrt_u**4)))
    sigmav_peak_arr.append(sigmav_peak)
#=========
    sigmav_taylor = (alpha**2*sqrt_u**2/M_ex**2)*(1+3/(2*x))*( 1/((sqrt_u**2-1)**2+epsilon)
        - (sqrt_u**2-1)*sqrt_u**2*(6/x)/(2*((sqrt_u**2-1)**2+epsilon)**2) )
    sigmav_taylor_arr.append(sigmav_taylor)
plt.plot(sqrt_u_arr, sigmav_num_arr, '--k', label=r'num',linewidth=2)
```

```
plt.plot(sqrt_u_arr, sigmav_taylor_arr, 'y', label='Taylor')
plt.plot(sqrt_u_arr, sigmav_0_arr, 'b', label=r'0')
plt.plot(sqrt_u_arr, sigmav_subs_arr, 'g', label=r'subs')
plt.plot(sqrt_u_arr, sigmav_peak_arr, 'r', label=r'peak')
plt.ylim(1e-8,0.1)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$\sqrt{u} = 2m_1/M_{ex}$")
plt.ylabel(r'$< \sigma v >$')
```

7 Figure 6

7.1 6a

```
epsilon = 7.5e-4
M_ex = 91
x = 25
alpha = 0.01
g_star = 106.75
MPl = 1.22e19
sqrt_u_arr = []
omega_0_arr = []
omega_subs_arr = []
omega_num_arr = []
for sqrt_u in np.arange(0.6,1.201,0.001):
    sqrt_u_arr.append(sqrt_u)
    J0 = (alpha**2/M_ex**2)*sqrt_u**2*(1+3/(4*x))*(1/x)*(1/((1-sqrt_u**2)**2+epsilon))
    omega_0 = (1.07e9/(np.sqrt(g_star)*MPl))/J0
    omega_0_arr.append(omega_0)
    Jsubs = ((alpha**2*sqrt_u**2/(1-3/(2*x)))/(M_ex**2*((1-sqrt_u**2/(1-3/(2*x)))**2+epsilon
        )))*(1-3/(2*x))/x
    omega_subs = (1.07e9/(np.sqrt(g_star)*MPl))/Jsubs
    omega_subs_arr.append(omega_subs)
    Jnum_integrand = lambda v : v*((alpha**2*sqrt_u**2/(1-v**4/2))/(M_ex**2*((1-sqrt_u
        **2/(1-v**2/4))**2+epsilon)))*erfc(v*np.sqrt(x)/2)
    Jnum_integral = quad(Jnum_integrand , 0,oo )
    Jnum = Jnum_integral[0]
    omega_num = (1.07e9/(np.sqrt(g_star)*MPl))/Jnum
    omega_num_arr.append(omega_num)
plt.plot(sqrt_u_arr, omega_num_arr, '--k', label=r'num',linewidth=2)
plt.plot(sqrt_u_arr, omega_0_arr, 'b', label=r'0')
plt.plot(sqrt_u_arr, omega_subs_arr, 'r', label=r'subs')
plt.ylim(1e-8,0.1)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$\sqrt{u} = 2m_1/M_{ex}$")
plt.ylabel(r'$\Omega_1h^2$')
```

7.2 6b

```
epsilon = 1e-6
M_ex = 91
x = 25
alpha = 0.01
g_star = 106.75
MPl = 1.22e19
sqrt_u_arr = []
```

```
omega_0_arr = []
omega_subs_arr = []
omega_num_arr = []
omega_peak_arr = []
for sqrt_u in np.arange(0.6,1.201,0.001):
    sqrt_u_arr.append(sqrt_u)
    J0 = (alpha**2/M_ex**2)*sqrt_u**2*(1+3/(4*x))*(1/x)*(1/((1-sqrt_u**2)**2+epsilon))
    omega_0 = (1.07e9/(np.sqrt(g_star)*MPl))/J0
    omega_0_arr.append(omega_0)
#=============================================================
    Jsubs = ((alpha**2*sqrt_u**2/(1-3/(2*x)))/(M_ex**2*((1-sqrt_u**2/(1-3/(2*x)))**2+epsilon
        )))*(1-3/(2*x))/x
    omega_subs = (1.07e9/(np.sqrt(g_star)*MPl))/Jsubs
    omega_subs_arr.append(omega_subs)
#=============================================================
    Jnum_integrand = lambda v : v*sqrt_u**2/(1-v**2/4)*erfc(v*np.sqrt(x)/2)/(((1-sqrt_u
        **2/(1-v**2/4))**2+epsilon))
    Jnum_integral = quad(Jnum_integrand , 0,oo )
    Jnum = Jnum_integral[0]*alpha**2/M_ex**2
    omega_num = (1.07e9/(np.sqrt(g_star)*MPl))/Jnum
    omega_num_arr.append(omega_num)
#=============================================================
    Jpeak = (4*sqrt_u**2/np.sqrt(epsilon))*(alpha**2/M_ex**2)*np.arctan(np.sqrt((1-sqrt_u
        **2)/(epsilon*sqrt_u**4)))*erfc(np.sqrt(x*(1-sqrt_u**2)))
    omega_peak = (1.07e9/(np.sqrt(g_star)*MPl))/Jpeak
    omega_peak_arr.append(omega_peak)
#=============================================================
plt.plot(sqrt_u_arr, omega_num_arr, '--k', label=r'$\Omega_{num}$', linewidth=2)
plt.plot(sqrt_u_arr, omega_0_arr, 'b', label=r'$\Omega_0$')
plt.plot(sqrt_u_arr, omega_subs_arr, 'g', label=r'$\Omega_{subs}$')
plt.plot(sqrt_u_arr, omega_peak_arr, 'r', label=r'$\Omega_{peak}$')
plt.ylim(1e-8,0.1)
plt.yscale("log")
plt.margins(0)
plt.legend()
plt.xlabel(r"$\sqrt{u} = 2m_1/M_{ex}$")
plt.ylabel(r'$\Omega_1h^2$')
```