Halo Model

Guilherme Pacheco Paredes - 12693754

Instituto de Física, Universidade de São Paulo - SP

Abstract

This model incorporates various components, including the halo mass function, halo bias, and density profiles, to provide a semi-analytic description of the matter power spectrum across both linear and nonlinear scales. In this project, we developed a computational approach to calculate the power spectrum using the halo model, comparing our results with those from the CAMB library for linear and nonlinear spectra. Through numerical integration methods implemented in Python, we computed the halo mass function, halo bias, and individual density profiles. Our analysis reveals a normalization of $g(\sigma) = 0.9928$ ***and*** $b(\nu) = 0.9959$***. The characteristic mass*** M^* ***corresponding to*** $\nu(M*) = 1$***, for*** $z = 0$ ***is*** $M^* = 1.91 \times 10^{12}\ M_\odot/h$ ***and for*** $z = 1$ ***is*** $M^* = 5.61 \times 10^{10} M_\odot/h$***. The results highlight the accuracy and limitations of the halo model in reproducing both linear and nonlinear power spectra and provide insight into the evolution of structure formation at different redshifts.***

I. Introduction

The formation of structures in the Universe is one of the most extensively studied topics in modern cosmology. Within this context, the halo model provides a robust theoretical framework for describing the distribution of dark matter and galaxies in the cosmos. It integrates various components, such as the halo mass function, halo bias, and density profiles, to construct a semi-analytic representation of the matter power spectrum, covering both linear and nonlinear scales.

In this project, we implemented a computational approach to calculate the power spectrum using the halo model, comparing it to the linear and nonlinear power spectra obtained with the CAMB library. The work involves computing the halo mass function, halo bias, and individual density profiles using numerical integration methods in Python. The final analysis evaluates the accuracy and limitations of the halo model in relation to predictions from linear and nonlinear theories.

II. Theory

I. Halo Mass-Function

In order to obtain the Halo Mass-Function, we have to compute first the variance σ^2 of linear fluctuations on a scale R

$$\sigma^2(z, R) = G^2(z) \int \frac{k^2 dk}{2\pi^2} |W(kR)|^2 P_L(k) \tag{1}$$

where $P_L(k)$ is the linear matter power spectrum at $z = 0$ that will be provided by `CAMB`,

$$W(kR) = \frac{3}{k^2R^2}\left[\frac{\sin(kr)}{kR} - \cos(kR)\right] \tag{2}$$

is the Fourier Transform of a spherical top-hat window of radius R and

$$G(z) = \frac{D(z)}{D(z=0)} = \frac{H(z)}{H_0}\frac{\int_z^\infty \frac{(1+z')}{H(z')^3}dz'}{\int_0^\infty \frac{(1+z')}{H(z')^3}dz'} \tag{3}$$

is the the linear Growth Factor of perturbations normalized to its value today in which $H(Z)$ Hubble Factor. After that we must compute by finite difference

$$\frac{d\ln\sigma^{-1}}{d\ln M} = -\frac{M}{\sigma}\frac{d\sigma}{dM} \tag{4}$$

where M is the mass and follows the relation

$$M = \frac{4\pi R^3 \bar{\rho}_{m0}}{3}, \tag{5}$$

in which $\bar{\rho}_{m0} = \Omega_m\ \rho_{\text{crit},0}$ and

$$\frac{dn(z,M)}{d\ln M} = g(\sigma)\frac{\bar{\rho}_{m0}}{M_{vir}}\frac{d\ln\sigma^{-1}}{d\ln M_v ir}, \tag{6}$$

with

$$g(\sigma) = B\left[\left(\frac{\sigma}{e}\right)^{-d} + \sigma^{-f}\right]\exp\left(-\frac{g}{\sigma^2}\right). \tag{7}$$

We must also assure that $g(\sigma)$ is properly normalized.

II. Halo Bias

To obtain the Halo Bias, using the fiducial cosmology (i.e $\Omega_k = 0$, $\Omega_m = 0.23$, $\Omega_r = 8 \times 10^{-8}$, $\Omega_{DE} = 1 - (\Omega_k + \Omega_m + \Omega_r)$, $\omega = -1$) and the fit from Tinker et al. 2010, we must compute the following expression

$$b(z, M) = 1 - A\frac{\nu^a}{\nu^a + \delta_c^a} + B\nu^b + C\nu^c \tag{8}$$

where $\nu = \delta_c/\sigma$ and $\delta_c = 1.686$. We must also assure that the bias function is properly normalized.

III. Halo Profile

Again using the fiducial cosmology, we intend to compute the Halo Profile for the fit of NFW

$$\rho(r|M_{vir}, z) = \frac{\rho_s}{cr/r_{vir}(1 + cr/r_{vir})^2} \tag{9}$$

for different values of $M_{vir} = M$ and redshifts, where the formulae will be used:

$$\rho_s = \frac{c^3 M_{vir}}{4\pi r_{vir}^3}\left[\frac{1}{\ln(1+c) - \frac{c}{1+c}}\right] \tag{10}$$

$$\Delta_c = \frac{3M_{vir}}{\rho_{\rm crit}(z)4\pi r_{vir}^3} = 18\pi^2 + 82x - 39x^2 \tag{11}$$

$$x = \omega_m(z) - 1 \tag{12}$$

$$\omega_m(z) = \frac{\Omega_m(1+z)^3}{E^2(z)} \tag{13}$$

$$E^2(z) = \Omega_m(1+z)^3 + \Omega_{DE}(1+z)^{3(1+\omega)} \tag{14}$$

$$\rho_{\rm crit}(z) = \rho_{\rm crit,0}E^2(z) \tag{15}$$

$$c(M_{vir}, z) = \frac{9}{1+z}\left(\frac{M_{vir}}{M*}\right), \quad \nu(M*) = 1 \tag{16}$$

Finally, we must compute numerically the Fourier Transform for the spherically symmetric Halo Profile at $z = 0$

$$u(k|M_{vir}) = \int_0^{r_{vir}} dr\ 4\pi r^2 \frac{\sin(kr)}{kr}\frac{\rho(r|M_{vir})}{M_{vir}} \tag{17}$$

and ensure that $u(k|M_{vir}) \to 1$ as $k \to 0$ for all values of M_{vir} that will be computed and compare it to the one obtained by Cooray & Sheth 2002.

IV. Halo Model: Nonlinear Power Spectrum

Finally, we will combine the Halo Mass-Function to the Halo Bias and the Halo Profile and obtain the Halo-Model power spectrum at $z = 0$ for

$$P^{1h} = \int d\ln M \frac{M^2}{\bar{\rho}_{m0}^2}\frac{dn}{d\ln M}|u(k|M)|^2 \tag{18}$$

and

$$P^{2h} = \left[\int d\ln M \frac{M}{\bar{\rho}_{m0}}\frac{dn}{d\ln M} b(M)u(k|M)\right]^2 P_L(K). \tag{19}$$

We will compare them to the linear power spectrum $P_L(k)$ from `CAMB` and the nonlinear power spectrum $P^{HF}(k)$ from the halofit code by Takahashi et al. 2012 that can also be obtained by `CAMB`.

III. Results and Discussion

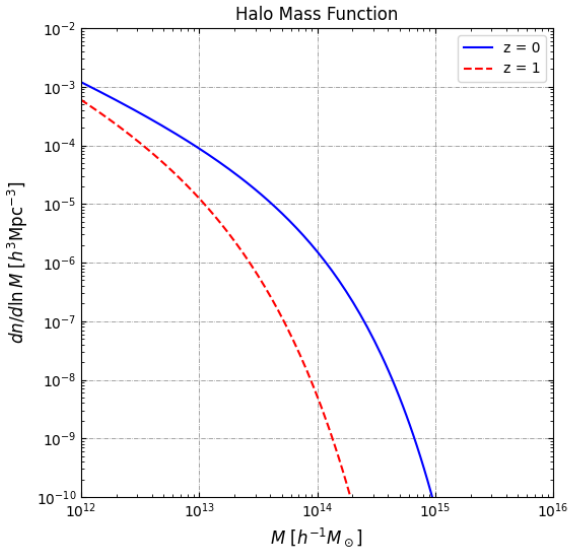

Figure 1: *Halo Mass-function for $z = 0, 1$, both in log-scale. The $g(\sigma)$ is computed following the values of parameter $\Delta = 400$ with $B = 0.484$, $d = 2.30$, $e = 0.93$, $f = 0.48$, $g = 1.403$ from Tinker et all. 2008*

The value obtained for the normalization of $g(\sigma)$ by the numerical computation of $\int[g(\sigma)\sigma]d\sigma$ is 0.99281. The relative difference is about 0.00724 and this non-exact normalization is due the limited range of σ values and therefore k values computed by `CAMB`. The normalization would only be exact for an infinite range of values, which

is not the case for the model. By Figure 1, the number of halos decreases as the redshift increases and is relatively high for small masses. This shows how smaller halos are less affected than the bigger halos, even though they also decrease significantly.

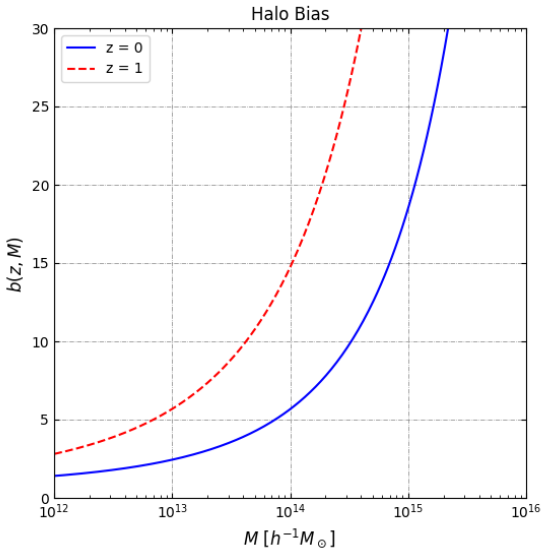

Figure 2: *Halo Bias for* $z = 0, 1$. *The bias function is computed following the values of the parameters* $A = 1 + 0,24ye^{-(4/y)^4}$, $a = 0.44y - 0.88$, $B = 0.183$, $b = 1.5$, $C = 0.019 + 0.0107y + 0.19e^{-(4/y)^4}$, $c = 2.4$, *with* $y = \log_{10} 400$ *from Tinker et all. 2010.*

The value obtained for the normalization of $b(\nu)$ by the numerical computation of $\int [g(\nu)b(\nu)/\nu]d\nu$ is 0.99586. The relative difference is about 0.00415 and this non-exact normalization, as for $g(\sigma)$, is due the limited range of ν values and therefore k values computed by `CAMB`. The normalization would only be exact for an infinite range of values, which is not the case for the model. By Figure 2 the curves increase as the mass increases and they increase more rapidly for bigger redshifts.

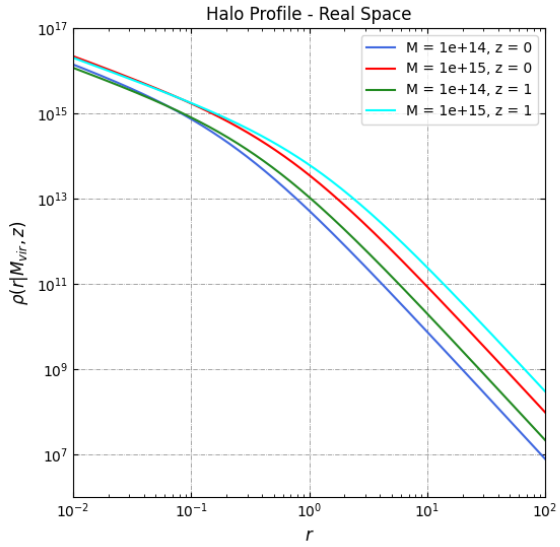

Figure 3: *The density distribution of matter with the Navarro-Frenk-White (NFW) model.*

The value of $M*$ obtained corresponding to $\nu(M*) = 1$, for $z = 0$ is $M^* = 1.91 \times 10^{12}\ M_\odot/\text{h}$ and for $z = 1$ is $M^* = 5.61 \times 10^{10} M_\odot/\text{h}$. By Figure 3 no much difference can be seen as it varies the mass and the redshift. Although, it is possible to see a rapidly decreasing behavior as the r values increase and a likely constant decrease for bigger values of r.

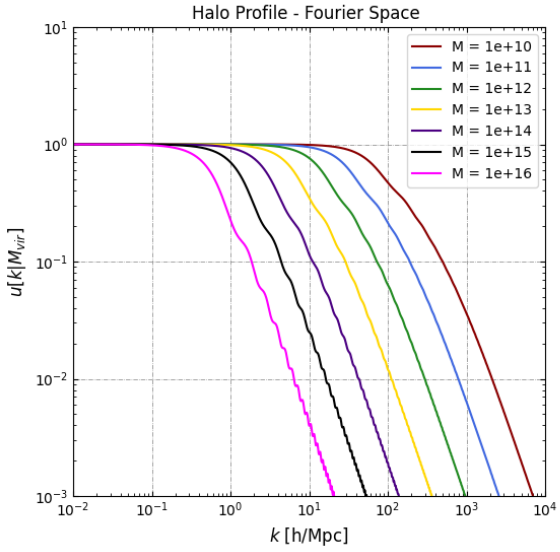

Figure 4: *The Numerical Fourier Transform describing the halo's contribution to the power spectrum.*

As expected by the normalization, $u(k|M_{vir} \rightarrow 1$ as $k \rightarrow 0$ and for bigger masses decaying more rapidly as k increases.

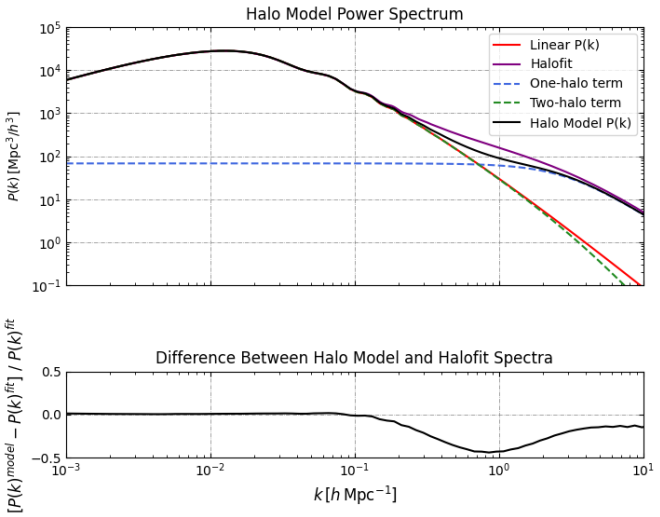

Figure 5: *The linear $P_L(k)$ and the non-linear from Halofit $P^{HF}(K)$ power spectrum from CAMB, the 1-Halo Term $P^{1h}(k)$ and the 2-Halo Term $P^{2h}(k)$ and their sum defining the Halo-Model on the top and the relative difference between the Halo-Model and Halofit spectra on the bottom.*

Also, as expected, as $k \rightarrow 0$ $u(k|M_{vir}) \rightarrow 1$ which can be seen as $P_L(k)$, the Halofit and the Halo-model evolve likely for small values of k.

IV. Conclusions

The Halo-Model obtained by numerical and analytical modulation appears satisfactory. First, as expected for the Halo Mass Function, the number of halos decreases rapidly for larger halo masses, as well as for smaller redshifts, which corresponds to the greater abundance of massive halos today. Additionally, the normalization of $g(\sigma)$ shows that the model performs well, even though it relies on finite values of k.

For the Halo Bias, the results align with theoretical expectations, where halos with higher masses exhibit a stronger bias relative to the underlying matter distribution. The bias increases with mass, particularly at higher redshifts, reflecting the fact that more massive halos are more strongly clustered.

Regarding the density profiles, the numerical implementation of the Navarro-Frenk-White (NFW) model for halo density shows a typical decline as the radius increases, consistent with the expected behavior of dark matter halos. The results match the theoretical predictions, with the density profile sharply decreasing with increasing radius, as expected for the NFW model, as can be seen below by the model from Cooray & Sheth 2002.:

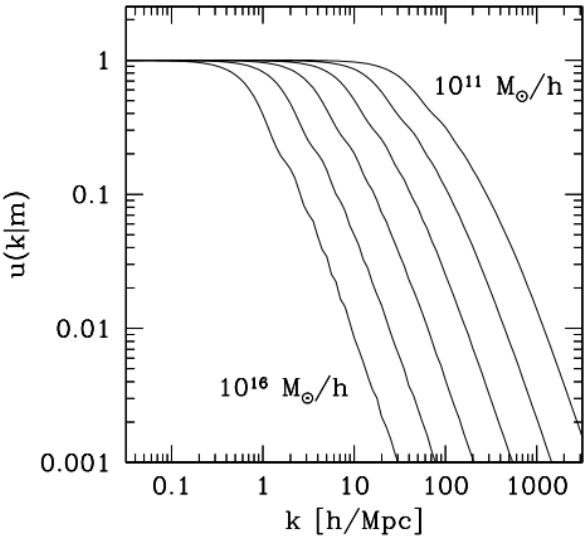

Figure 6: *The density distribution of matter with the Navarro-Frenk-White (NFW) model from Cooray & Sheth 2002.*

Furthermore, the numerical Fourier transform of the halo model was successfully computed, and the resulting power spectrum was found to be in excellent agreement with the expected behavior, both for the linear and nonlinear regimes. The power spectrum analysis shows that the halo model, despite its simplifications, provides a good representation of the matter distribution, even at smaller scales where nonlinear effects become more significant.

Overall, while the model does have certain limitations, such as the reliance on certain approximations and finite k-values, the results demonstrate that it successfully captures the essential features of structure formation and halo clustering. The comparison with the power spectra obtained from CAMB confirms the validity of the halo model, making it a valuable tool for cosmological studies.

V. Appendix

The program implemented in this project:

```
'''
Halo-Mass Function
'''

import camb
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import trapezoid, quad
from scipy.interpolate import InterpolatedUnivariateSpline, interp1d

# Constants
H0 = 67.5
Om_m = 0.23
ombh2 = 0.022
omch2 = Om_m * (H0 / 100) ** 2 - ombh2
Om_r = 8e-5
Om_DE = 1 - (Om_m + Om_r)
w = -1
rho_crit_0 = 2.775 * (H0 / 100) ** 2 * 1e11
rho_m0 = Om_m * rho_crit_0
delta_c = 1.686

# Function to get the power spectrum P(k) by CAMB parameters
def P_L():
    pars = camb.set_params(H0=H0, ombh2=ombh2, omch2=omch2, omk=0, TCMB=2.7255)
    pars.set_matter_power(redshifts=[0], kmax=100)
    pars.set_dark_energy(w=w)
    results = camb.get_results(pars)
    k_lin, z_lin, pk_lin = results.get_matter_power_spectrum(minkh=1e-4, maxkh=100, npoints=1000)
    return k_lin, z_lin, pk_lin


# Window function W(k, R)
def W(k, R):
    return (3 / (k * R) ** 2) * (np.sin(k * R) / (k * R) - np.cos(k * R))


# Hubble function H(z)
def H(z):
    return H0 * np.sqrt(Om_m * (1 + z) ** 3 + Om_r * (1 + z) ** 4 + Om_DE * (1 + z) ** (3 * (1 + w)))


# Growth factor G(z) normalized by D(z) / D(z=0)
def G(z):
    integrand = lambda z_prime: (1 + z_prime) / H(z_prime) ** 3
    integral, _ = quad(integrand, z, np.inf)
    integral_0, _ = quad(integrand, 0, np.inf)
    return H(z) / H0 * integral / integral_0


# Function to calculate sigma(z, R)
def sigma_zR(zet, R, k, p_l):
    G_z = G(zet)
    integrand = (k ** 2) * (W(k, R) ** 2) * p_l[0, :]
    integral_value = trapezoid(integrand, k)
    return G_z * np.sqrt(integral_value / (2 * np.pi ** 2))


# Combined function: calculates sigma, its derivatives, and interpolates
```

```
def calculate_interpolation(R_value, k, p_l, zet_values):
    M_vir = (4 / 3) * np.pi * R_value ** 3 * rho_m0

    # Calculate sigma for each R and z value
    sigma_z = np.array([[sigma_zR(zet, R_value, k, p_l) for R_value in R_values] for zet in zet_values])

    # Interpolated sigma and derivatives
    splines_sigma = []
    splines_dln_sigma_inv_dln_M = []

    for sigma in sigma_z:
        # Interpolated sigma
        spline_sigma = InterpolatedUnivariateSpline(M_vir, sigma, k=3)
        splines_sigma.append(spline_sigma)

        # Calculate derivative d ln ^{-1} / d ln M
        d_sigma_dM = np.gradient(sigma, M_vir)
        dln_sigma_inv_dln_M = -M_vir / sigma * d_sigma_dM

        # Interpolated d ln ^{-1} / d ln M
        spline_dln_sigma_inv_dln_M = InterpolatedUnivariateSpline(M_vir, dln_sigma_inv_dln_M, k=3)
        splines_dln_sigma_inv_dln_M.append(spline_dln_sigma_inv_dln_M)


    return M_vir, sigma_z, splines_sigma, splines_dln_sigma_inv_dln_M


# Function g(sigma) by Tinker et al. 2008
def g_func(sigma):
    B, d, e, f, g = 0.494, 2.30, 0.93, 0.48, 1.403

    sigma = np.asarray(sigma)
    return B * ((sigma / e) ** (-d) + sigma**(-f)) * np.exp(-g / (sigma ** 2))



# Call for power spectrum function
k_value, z_value, pk_value = P_L()

R_values = np.logspace(-2, 8, 1000)
z_values = [0, 1]

# Compute dn/dlnM for each z value
dn_dlnM_curves = []

# Loop principal
for z in z_values:
    M_values, sigma_z, splines_sigma, splines_dln_sigma_inv_dln_M = \
        calculate_interpolation(R_values, k_value, pk_value, [z])
    sigma_at_z = np.asarray(sigma_z[0])

    dn_dlnM = g_func(sigma_at_z) * rho_m0 / M_values * \
                splines_dln_sigma_inv_dln_M[0](M_values)

    dn_dlnM_curves.append((M_values, dn_dlnM))

# Plot dn/dlnM versus M for z = 0 and z = 1
f = plt.figure(figsize=(6, 6))
ax = f.add_subplot(111)

for z, (M_values, dn_dlnM_g) in zip(z_values, dn_dlnM_curves):
    if z == 0:
        plt.loglog(M_values, dn_dlnM_g, label=f'z = {z}', color='blue', linestyle='-')
```

```
    else:
        plt.loglog(M_values, dn_dlnM_g, label=f'z = {z}', color='red', linestyle='--')



plt.grid(visible=True, color='gray', linestyle='-.', linewidth=0.5)
ax.xaxis.set_tick_params(direction='in',  which='both', top=True, bottom=True)
ax.yaxis.set_tick_params(direction='in',  which='both', left=True, right=True)
plt.xlim(1e12, 1e16)
plt.ylim(1e-10, 1e-2)
plt.xlabel(r'$M \ [h^{-1}M_\odot]$', fontsize=12)
plt.ylabel(r'$dn/d\ln M \  [h^3 \mathrm{Mpc}^{-3}]$', fontsize=12)
plt.title('Halo Mass Function')
plt.legend()
plt.show()


'''
Halo Bias
'''

# The Bias Function
def b_z(nu):
    y = np.log10(400)
    A = 1 + 0.24 * y * np.exp(-(4 / y) ** 4)
    a = 0.44 * y - 0.88
    B = 0.183
    b = 1.5
    C = 0.019 + 0.107 * y + 0.19 * np.exp(-(4 / y) ** 4)
    c = 2.4

    return 1 - A * nu ** a / (nu ** a + delta_c ** a) + B * nu ** b + C * nu ** c



f = plt.figure(figsize=(6, 6))
ax = f.add_subplot(111)

# Loop principal
for z in z_values:
    M_vir, sigma_z, splines_sigma, splines_dln_sigma_inv_dln_M = calculate_interpolation(R_values, kh, pk, [z])
    sigma_at_z = sigma_z[0]
    nu = delta_c / sigma_at_z
    bias_values = b_z(nu)
    if z == 0:
        plt.plot(M_vir, bias_values, label=f'z = {z}', color='blue', linestyle='-')
    else:
        plt.plot(M_vir, bias_values, label=f'z = {z}', color='red', linestyle='--')


plt.xscale('log')
plt.yscale('linear')
plt.grid(visible=True, color='gray', linestyle='-.', linewidth=0.5)
ax.xaxis.set_tick_params(direction='in',  which='both', top=True, bottom=True)
ax.yaxis.set_tick_params(direction='in',  which='both', left=True, right=True)
plt.xlim(1e12, 1e16)
plt.ylim(0, 30)
plt.xlabel(r'$M \ [h^{-1}M_\odot]$', fontsize=12)
plt.ylabel(r'$b(z, M)$ ', fontsize=12)
plt.title(r'Halo Bias')
plt.legend()
plt.show()
```

```
'''
Halo Profile
'''

# Function to find nu(M*) = 1
def find_M_star(sigma_crit, kh, pk, z, R_values):
    sigma_R = np.array([sigma_zR(z, R, kh, pk) for R in R_values])
    R_crit = R_values[np.argmin(np.abs(sigma_R - sigma_crit))]
    M_star = (4 / 3) * np.pi * R_crit ** 3 * rho_m0

    print(f"The value of M corresponding to sigma(z=0, M)  1.686"
          f" is approximately {M_star:.2e} M_sun/h.")
    return M_star

# Function rho_NFW
def rho(rho_s, c, r_values, r_vir):
    return rho_s / (c * r_values / r_vir) * (1 / (1 + c * r_values / r_vir) ** 2)


def E2(zet):
    return Om_m * (1 + zet)**3 + Om_DE * (1 + zet)**(3 * (1 + w))


def rho_cri(zet):
    return rho_crit_0 * E2(zet)


def w_m(zet):
    return Om_m * (1 + zet) ** 3 / E2(zet)


def Delta_c(zet):
    x = w_m(zet) - 1
    return 18 * np.pi ** 2 + 82 * x - 39 * x ** 2


def r_vir(zet, M_vir):
    return ((3 * M_vir) / (4 * np.pi * Delta_c(zet) * rho_cri(zet))) ** (1/3)


def c(M_vir, M_star, zet):
    return 9 / (1 + zet) * (M_vir / M_star) ** (-0.13)


def rho_s(M_vir, M_star, zet):
    c_value = c(M_vir, M_star, zet)
    return (M_vir / (4 * np.pi * r_vir(zet, M_vir) ** 3)) * (c_value **3 / (np.log(1 + c_value) - c_value / (1 + c_value)))


# Call for power spectrum function
kh, zs, pk = P()

R_values = np.logspace(-2, 8, 1000)
z_values = [0, 1]
r_values = np.logspace(-6, 2, 1000)
M_vir_values1 = [1e14, 1e15]

f = plt.figure(figsize=(6, 6))
ax = f.add_subplot(111)
colors_1 = ['royalblue', 'red', 'forestgreen', 'cyan']  # Lista de cores

for i, z in enumerate(z_values):
    for j, M_vir in enumerate(M_vir_values1):
```

```
        M_star = find_M_star(1.686, kh, pk, z, R_values)
        rho_s_value = rho_s(M_vir, M_star, z)
        r_vir_value = r_vir(z, M_vir)
        c_value = c(M_vir, M_star, z)
        rho_values = rho(rho_s_value, c_value, r_values, r_vir_value)
        color = colors_1[i * len(M_vir_values1) + j]
        plt.loglog(r_values, rho_values, label=f'M = {M_vir:.0e}, z = {z}', color=color)


plt.grid(visible=True, color='gray', linestyle='-.', linewidth=0.5)
ax.xaxis.set_tick_params(direction='in',  which='both', top=True, bottom=True)
ax.yaxis.set_tick_params(direction='in',  which='both', left=True, right=True)

plt.xlim(1e-2, 1e2)
plt.ylim(1e6, 1e17)

plt.xlabel(r'$r$', fontsize=12)
plt.ylabel(r'$\rho(r|M_{vir},z)$', fontsize=12)
plt.legend(loc='best')
plt.title('Halo Profile - Real Space')
plt.show()


M_vir_values2 = [1e10, 1e11, 1e12, 1e13, 1e14, 1e15, 1e16]


# Function to calculate u(k|M_vir) numerically
def u_numeric(k, M_vir, r_vir, rho_s, c):
    rs = r_vir / c
    prefactor = 4 * np.pi * rho_s * rs ** 3 / M_vir

    def integrand(r):
        rho_r = rho(rho_s, c, r, r_vir)
        return np.sin(k * r) / (k * r) * rho_r * r ** 2

    integral_value, _ = quad(integrand, 0, r_vir)

    return prefactor * integral_value

k_values = kh
z = 0
colors_2 = ['darkred', 'royalblue', 'forestgreen', 'gold', 'indigo', 'black', 'magenta', 'teal', 'orchid', 'darkviolet']
f = plt.figure(figsize=(6, 6))
ax = f.add_subplot(111)

for i, M_vir in enumerate(M_vir_values2):
    M_star = find_M_star(1.686, kh, pk, z, R_values)
    rho_s_value = rho_s(M_vir, M_star, z)
    r_vir_value = r_vir(z, M_vir)
    c_value = c(M_vir, M_star, z)

    u_values_num = []

    for k in k_values:
        u_values_num.append(u_numeric(k, M_vir, r_vir_value, rho_s_value, c_value))

    plt.loglog(k_values, u_values_num, label=f'M = {M_vir:.0e}', color=colors_2[i])


plt.grid(visible=True, color='gray', linestyle='-.', linewidth=0.5)
ax.xaxis.set_tick_params(direction='in',  which='both', top=True, bottom=True)
ax.yaxis.set_tick_params(direction='in',  which='both', left=True, right=True)
plt.xlim(1e-2, 1e4)
```

```
plt.ylim(1e-3, 1e1)
plt.xlabel('$k$ [h/Mpc]', fontsize=12)
plt.ylabel(r'$u[k|M_{vir}]$', fontsize=12)
plt.title('Halo Profile - Fourier Space ')
plt.legend(loc='best')
plt.show()

'''
Halo Model: Nonlinear Power Spectrum
'''


# Non-linear Halofit power spectrum by CAMB
def P_hf():
    pars = camb.set_params(H0=H0, ombh2=ombh2, omch2=omch2, omk=0, TCMB=2.7255)
    pars.set_matter_power(redshifts=[0], kmax=1500)
    pars.set_dark_energy(w=w)
    results = camb.get_results(pars)
    pars.NonLinearModel.set_params(halofit_version='takahashi')
    kh_taka, z_taka, pk_taka = results.get_nonlinear_matter_power_spectrum(params=pars)
    return kh_taka, z_taka, pk_taka


# Integrals for Pk terms
def pk_1term(M_vir, splines_sigma, splines_dln_sigma_inv_dln_M, u_values):
    sigma_values = splines_sigma[0](M_vir)
    dln_sigma_inv_dln_M_values = splines_dln_sigma_inv_dln_M[0](M_vir)

    dn_dlnM = g_func(sigma_values) * rho_m0 / M_vir * dln_sigma_inv_dln_M_values

    integrand = (M_vir / rho_m0)**2 * dn_dlnM * (np.abs(u_values)) ** 2

    pk_1 = trapezoid(integrand, np.log(M_vir))

    return pk_1

def pk_2term(M_vir, splines_sigma, splines_dln_sigma_inv_dln_M, u_values, pk):
    sigma_values = splines_sigma[0](M_vir)
    dln_sigma_inv_dln_M_values = splines_dln_sigma_inv_dln_M[0](M_vir)

    dn_dlnM = g_func(sigma_values) * rho_m0 / M_vir * dln_sigma_inv_dln_M_values

    nu = delta_c / sigma_values
    bias = b_z(nu)

    integrand_1 = (M_vir / rho_m0) * bias * dn_dlnM
    integrand_2 = (M_vir / rho_m0) * bias * u_values * dn_dlnM

    # Normalizing for small values of k with u(k,M) = 1
    A = trapezoid(integrand_1, np.log(M_vir))
    pk_2 = trapezoid(integrand_2, np.log(M_vir))

    return (pk_2 / A) ** 2 * pk


# Call for linear and non-linear power spectrum function
kh, zs, pk = P()
kh_nlin, z_nlin, pk_nlin = P_hf()

R_values = np.logspace(-2, 8, 1000)
z = 0
r_values = np.logspace(-6, 5, 1000)
M_star = find_M_star(1.686, kh, pk, z, R_values)
```

```
pk_1term_values = np.zeros_like(kh)
pk_2term_values = np.zeros_like(kh)
p_tot_values = np.zeros_like(kh)

M_vir, sigma_z, splines_sigma, splines_dln_sigma_inv_dln_M = calculate_interpolation(R_values, kh, pk)
rho_s_value = rho_s(M_vir, M_star, z)
r_vir_value = r_vir(z, M_vir)
c_value = c(M_vir, M_star, z)

u_values = u_k_Mvir(kh, M_vir, r_vir_value, rho_s_value, c_value)

for i, u in enumerate(u_values):
    pk_1term_values[i] = pk_1term(M_vir, splines_sigma, splines_dln_sigma_inv_dln_M, u)
    pk_2term_values[i] = pk_2term(M_vir, splines_sigma, splines_dln_sigma_inv_dln_M, u, pk[0, i])
    p_tot_values[i] = pk_1term_values[i] + pk_2term_values[i]

f, axs = plt.subplots(2, 1, figsize=(8, 6),
                      gridspec_kw={'height_ratios': [6, 2], 'hspace': 0.5},
                      sharex=True)

# First plot: Halo Model, linear and non-linear Power Spectrum
axs[0].loglog(kh, pk[0, :], label="Linear P(k)", color='red')
axs[0].loglog(kh_nlin, pk_nlin[0, :], label="Halofit", color='purple')
axs[0].loglog(kh, pk_1term_values, label="One-halo term", color='royalblue', linestyle='--')
axs[0].loglog(kh, pk_2term_values, label="Two-halo term", color='forestgreen', linestyle='--')
axs[0].loglog(kh, p_tot_values, label="Halo Model P(k)", color='black')
axs[0].grid(visible=True, color='gray', linestyle='-.', linewidth=0.5)
axs[0].xaxis.set_tick_params(direction='in',  which='both', top=True, bottom=True)
axs[0].yaxis.set_tick_params(direction='in',  which='both', left=True, right=True)
axs[0].set_xlim(1e-3, 1e1)
axs[0].set_ylim(1e-1, 1e5)
axs[0].set_ylabel(r"$P(k) \, [\mathrm{Mpc}^3/h^3]$")
axs[0].set_title('Halo Model Power Spectrum')
axs[0].legend()


from scipy.interpolate import interp1d

# Interpolation values for the second plot
interpolated_pk_nlin = interp1d(kh_nlin, pk_nlin[0, :], kind='linear', bounds_error=False, fill_value="extrapolate")
pk_nlin_interpolated = interpolated_pk_nlin(kh)
difference = (p_tot_values - pk_nlin_interpolated) / pk_nlin_interpolated

# Second plot for relative difference between Halofit and Halo Model
axs[1].semilogx(kh, difference, color='black')
axs[1].set_ylim(-0.5, 0.5)
axs[1].grid(visible=True, color='gray', linestyle='-.', linewidth=0.5)
axs[1].xaxis.set_tick_params(direction='in',  which='both', top=True, bottom=True)
axs[1].yaxis.set_tick_params(direction='in',  which='both', left=True, right=True)
axs[1].set_xlabel(r"$k \, [h \, \mathrm{Mpc}^{-1}]$", fontsize=12)
axs[1].set_ylabel(r"$[P(k)^{model} - P(k)^{fit}] \  / \ P(k)^{fit}$", fontsize=12)
axs[1].set_title("Difference Between Halo Model and Halofit Spectra")

plt.tight_layout()
plt.show()
```

References

[1] Tinker, J. L., Kravtsov, A. V., Klypin, A., Abazajian, K., Warren, M., Yepes, G., Gottlöber, S., & Holz, D. E. (2008). Toward a halo mass function for precision cosmology: The limits of universality.

[2] Tinker, J. L., Robertson, B. E., Kravtsov, A. V., et al. (2010). The halo mass function at high redshift.

[3] Bryan, G. L., & Norman, M. L. (1997). Statistical prop-

erties of the mass function in hierarchical models.

[4] Cooray, A., & Sheth, R. (2002). Halo models of large scale structure.

[5] Takahashi, R., Sato, M., Nishimichi, T., Taruya, A., & Oguri, M. (2012). Revising the halofit model for the nonlinear matter power spectrum.