Additional_Plots

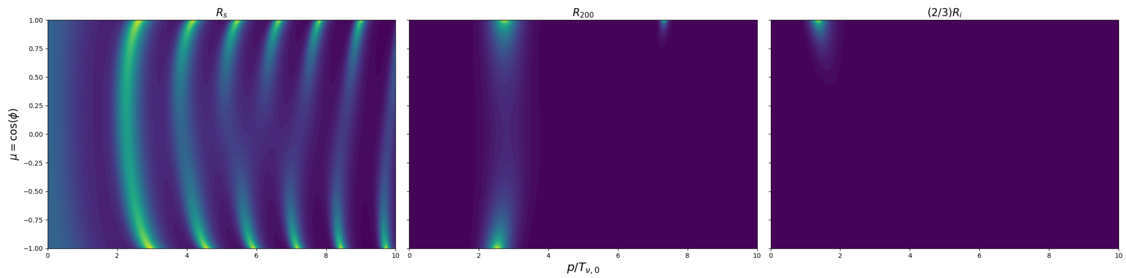
September 21, 2024

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from src.input import *
from src.distributions import f_FD
from src.units import UNITS
from scipy.interpolate import interp1d
from matplotlib import cm
from matplotlib.colors import Normalize
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
from scipy.stats import linregress
```

0.0.1 Orbits and Contours

[2]:
```
fig, axes = plt.subplots(1, 3, figsize=(24, 6), sharex=True, sharey=True)
mu_integrands = np.load('paper_plots_notebook/mu_integrands.npy')

mu_arr = np.cos(np.linspace(0, np.pi, 50))
extended_p_arr = np.geomspace(0.01, 50, 1000)
X_1, Y_1 = np.meshgrid(mu_arr, extended_p_arr)

contour = axes[0].contourf(Y_1, X_1, mu_integrands[0, :, :], levels=50,␣
  ↪cmap='viridis')
axes[0].set_ylabel(r'$\mu = \cos(\phi)$', fontsize = 16)
axes[0].set_xlim(0, 20)
axes[0].set_title(r'$R_s$', fontsize = 16)

contour = axes[1].contourf(Y_1, X_1, mu_integrands[1, :, :], levels=50,␣
  ↪cmap='viridis')
axes[1].set_xlabel(r'$p / T_{\nu,0}$', fontsize=18)
axes[1].set_xlim(0, 10)
axes[1].set_title(r'$R_{200}$', fontsize = 16)

contour = axes[2].contourf(Y_1, X_1, mu_integrands[2, :, :], levels=50,␣
  ↪cmap='viridis')
axes[2].set_xlim(0, 10)
axes[2].set_title(r'(2/3)$R_i$', fontsize = 16)
```

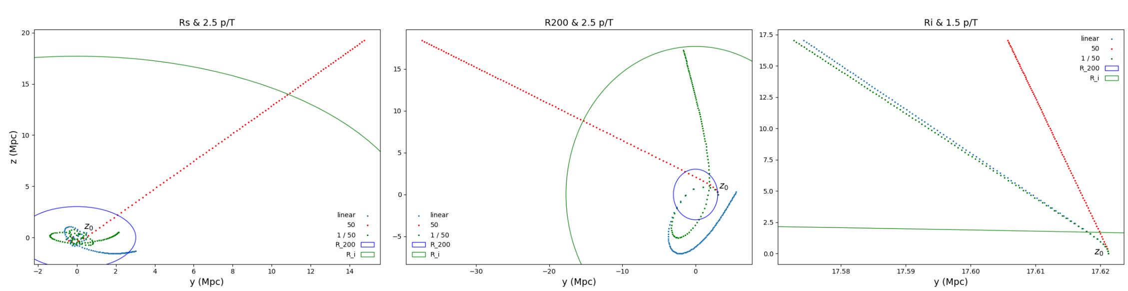
```python
plt.tight_layout()
plt.savefig('plot_save_vehicle.png')
plt.show()
```



[3]:

```python
### Chose momenta and mu values where the integrands peak, as indicated above ###
## So all are mu = 1. First two are p/T_nu = 2.5, and Ri is p/T_nu = 1.5
fig, axes = plt.subplots(1, 3, figsize=(24, 6))
positions_mod_funcs = np.load('paper_plots_notebook/positions_mod_funcs.npy')

# Subplot 1
axes[0].scatter(positions_mod_funcs[0,0,:,0], positions_mod_funcs[0,0,:,1], s=2, label = 'linear')
axes[0].scatter(positions_mod_funcs[0,1,:,0], positions_mod_funcs[0,1,:,1], s=2, color = 'red', label = '50')
axes[0].scatter(positions_mod_funcs[0,2,:,0], positions_mod_funcs[0,2,:,1], s=2, color = 'green', label = '1 / 50')
circle = plt.Circle((0, 0), radius=3.026, alpha=0.9, edgecolor='blue', facecolor='none', label='R_200')
circ = plt.Circle((0, 0), radius=17.7, alpha=0.9, edgecolor='green', facecolor='none', label='R_i')
axes[0].add_artist(circle)
axes[0].add_artist(circ)
axes[0].set_xlabel('y (Mpc)', fontsize=14)
axes[0].set_ylabel('z (Mpc)', fontsize=14)
axes[0].legend(markerfirst = False, frameon = False)
axes[0].set_title('Rs & 2.5 p/T', fontsize=14)
axes[0].text(0.36, 0.8, '$z_0$', fontsize=14)

# Subplot 2
axes[1].scatter(positions_mod_funcs[1,0,:,0], positions_mod_funcs[1,0,:,1], s=2, label = 'linear')
axes[1].scatter(positions_mod_funcs[1,1,:,0], positions_mod_funcs[1,1,:,1], s=2, color = 'red', label = '50')
```

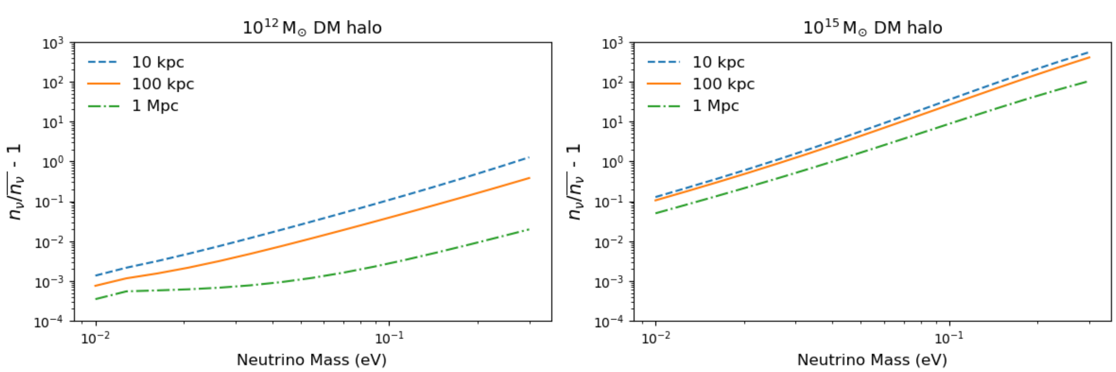
```
axes[1].scatter(positions_mod_funcs[1,2,:,0], positions_mod_funcs[1,2,:,1], s=2, color = 'green', label = '1 / 50')
circle = plt.Circle((0, 0), radius=3.026, alpha=0.9, edgecolor='blue', facecolor='none', label='R_200')
circ = plt.Circle((0, 0), radius=17.7, alpha=0.9, edgecolor='green', facecolor='none', label='R_i')
axes[1].add_artist(circle)
axes[1].add_artist(circ)
axes[1].set_xlabel('y (Mpc)', fontsize=14)
axes[1].legend(frameon = False)
axes[1].set_title('R200 & 2.5 p/T', fontsize=14)
axes[1].text(3.2, 0.7, '$z_0$', fontsize=14)

# Subplot 3
axes[2].scatter(positions_mod_funcs[2,0,:,0], positions_mod_funcs[2,0,:,1], s=2, label = 'linear')
axes[2].scatter(positions_mod_funcs[2,1,:,0], positions_mod_funcs[2,1,:,1], s=2, color = 'red', label = '50')
axes[2].scatter(positions_mod_funcs[2,2,:,0], positions_mod_funcs[2,2,:,1], s=2, color = 'green', label = '1 / 50')
circle = plt.Circle((0, 0), radius=3.026, alpha=0.9, edgecolor='blue', facecolor='none', label='R_200')
circ = plt.Circle((0, 0), radius=17.7, alpha=0.9, edgecolor='green', facecolor='none', label='R_i')
axes[2].add_artist(circle)
axes[2].add_artist(circ)
axes[2].set_xlabel('y (Mpc)', fontsize=14)
axes[2].legend(markerfirst = False, frameon = False)
axes[2].set_title('Ri & 1.5 p/T', fontsize=14)
axes[2].text(17.619, -0.1, '$z_0$', fontsize=14)

plt.tight_layout()
plt.savefig('plot_save_vehicle.png')
plt.show()
```



[ ]:

0.0.2 Mass-dependence of Neutrino Overdensity

[4]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
mass_overdens = np.load('paper_plots_notebook/mass_overdens.npy')
mass_arr = np.geomspace(0.01, 0.3, 15) #eV

ax1.loglog(mass_arr, mass_overdens[0,5,:] - 1, label = '10 kpc', linestyle =␣
 ↪'dashed')
ax1.loglog(mass_arr, mass_overdens[0,10,:] - 1, label = '100 kpc', linestyle =␣
 ↪'solid')
ax1.loglog(mass_arr, mass_overdens[0,14,:] - 1, label = '1 Mpc', linestyle =␣
 ↪'dashdot')
ax1.set_xlabel('Neutrino Mass (eV)', fontsize = 12)
ax1.set_ylabel(r'$n_{\nu} / \overline{n_{\nu}\,}$ - 1', fontsize=14)
ax1.set_title(r'$10^{12} \, \text{M}_{\odot}$ DM halo', fontsize = 13)
ax1.set_ylim(1e-4,1e3)
ax1.legend(loc = 'upper left', frameon = False, fontsize = 12)

ax2.loglog(mass_arr, mass_overdens[1,5,:] - 1, label = '10 kpc', linestyle =␣
 ↪'dashed')
ax2.loglog(mass_arr, mass_overdens[1,10,:] - 1, label = '100 kpc', linestyle =␣
 ↪'solid')
ax2.loglog(mass_arr, mass_overdens[1,14,:] - 1, label = '1 Mpc', linestyle =␣
 ↪'dashdot')
ax2.set_xlabel('Neutrino Mass (eV)', fontsize = 12)
ax2.set_ylabel(r'$n_{\nu} / \overline{n_{\nu}\,}$ - 1', fontsize=14)
ax2.set_title(r'$10^{15} \, \text{M}_{\odot}$ DM halo', fontsize = 13)
ax2.set_ylim(1e-4,1e3)
ax2.legend(frameon = False, fontsize = 12)

plt.tight_layout()
plt.savefig('plot_save_vehicle.pdf')
plt.show()
```



[5]:
```
# Calculate the power law index for both halo masses with linear regression
for mass_idx in range(2):
    data = mass_overdens[mass_idx, 5, :] - 1  # At 10 kpc
    log_mass = np.log10(mass_arr)
    log_data = np.log10(data)
    slope, _, _, _, _ = linregress(log_mass, log_data)
    print(f"Power law index for 10 kpc for 10^{12 + 3 * mass_idx} MSun: {slope:.
↪2f}")
```

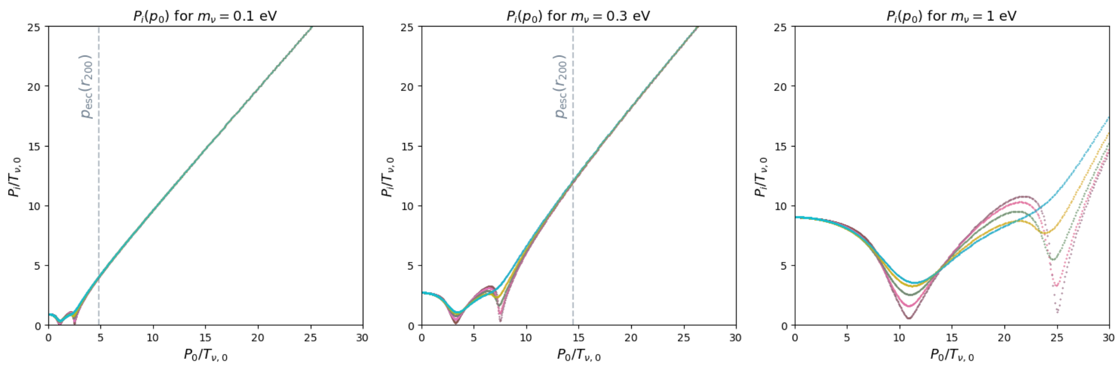
```
Power law index for 10 kpc for 10^12 MSun: 2.00
Power law index for 10 kpc for 10^15 MSun: 2.51
```

[ ]:
```

```

0.0.3 $P_i(p_0)$

[6]:
```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))
po_pi_data = np.load('paper_plots_notebook/po_pi_array.npy')
po_pi_arr = np.geomspace(0.01, 40, 2000)

for i in range(10):
    ax1.scatter(po_pi_arr, po_pi_data[0,i,:] * 0.1 * UNITS.eV / Tnu_0, s=0.1)
ax1.set_xlim(0, 30)
ax1.set_ylim(0, 25)
ax1.set_xlabel(r'$P_0 / T_{\nu,0}$', fontsize=13)
ax1.set_ylabel(r'$P_i / T_{\nu,0}$', fontsize=13)
ax1.set_title(r'$P_i(p_0)$ for $m_{\nu} = 0.1$ eV', fontsize=13)
ax1.vlines(4.818, 0, 30, linestyle='dashed', alpha=0.5, color='slategrey')
ax1.text(2.818, 20, r'$p_{\text{esc}}(r_{200})$', rotation=90,␣
↪verticalalignment='center', horizontalalignment='left', fontsize=14,␣
↪color='slategrey')


for i in range(10):
    ax2.scatter(po_pi_arr, po_pi_data[1,i,:] * 0.3 * UNITS.eV / Tnu_0, s=0.1)
ax2.set_xlabel(r'$p / T_{\nu,0}$', size=12)
ax2.set_xlim(0, 30)
ax2.set_ylim(0, 25)
ax2.set_xlabel(r'$P_0 / T_{\nu,0}$', fontsize=13)
ax2.set_ylabel(r'$P_i / T_{\nu,0}$', fontsize=13)
ax2.set_title(r'$P_i(p_0)$ for $m_{\nu} = 0.3$ eV', fontsize=13)
ax2.vlines(14.45, 0, 30, linestyle='dashed', alpha=0.5, color='slategrey')
ax2.text(12.45, 20, r'$p_{\text{esc}}(r_{200})$', rotation=90,␣
↪verticalalignment='center', horizontalalignment='left', fontsize=14,␣
↪color='slategrey')
```

```python
for i in range(10):
    ax3.scatter(po_pi_arr, po_pi_data[2,i,:] * 1 * UNITS.eV / Tnu_0, s=0.1)
ax3.set_xlabel(r'$p / T_{\nu,0}$', size=12)
ax3.set_xlim(0, 30)
ax3.set_ylim(0, 25)
ax3.set_xlabel(r'$P_0 / T_{\nu,0}$', fontsize=13)
ax3.set_ylabel(r'$P_i / T_{\nu,0}$', fontsize=13)
ax3.set_title(r'$P_i(p_0)$ for $m_{\nu} = 1$ eV', fontsize=13)
ax3.vlines(48.18, 0, 30, linestyle = 'dashed', alpha = 0.5)
plt.tight_layout()
#plt.savefig('plot_save_vehicle.pdf')
```



[ ]:

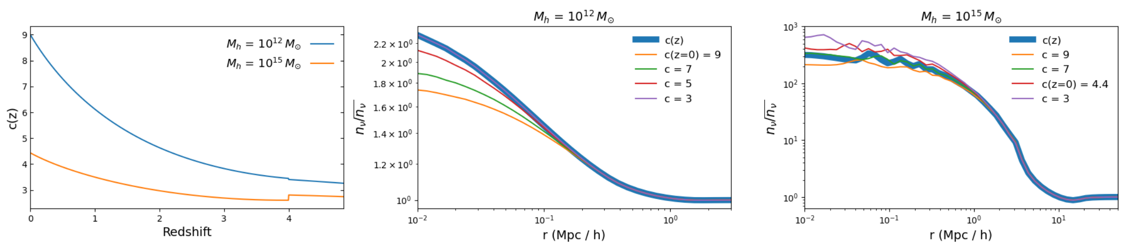
0.0.4 Concentration parameter

[8]:
```python
def concentration(redshift_array, Mh):
    # For redshift >= 4
    cond = redshift_array >= 4
    conc_high_z = 10 ** (
        1.3081
        - 0.1078 * (1 + redshift_array)
        + 0.00398 * (1 + redshift_array) ** 2
        + (0.0223 - 0.0944 * (1 + redshift_array) ** (-0.3907))
        * np.log10(Mh / UNITS.MSun)
    )

    # For redshift < 4
    conc_low_z = 10 ** (
        1.7543
        - 0.2766 * (1 + redshift_array)
```

```
        + 0.02039 * (1 + redshift_array) ** 2
        + (0.2753 + 0.00351 * (1 + redshift_array) - 0.3038 * (1 +␣
↪redshift_array) ** 0.0269)
        * np.log10(Mh / UNITS.MSun)
        * (
            1.0
            + (-0.01537 + 0.02102 * (1 + redshift_array) ** (-0.1475))
            * (np.log10(Mh / UNITS.MSun)) ** 2
        )
    )

    conc = np.where(cond, conc_high_z, conc_low_z)
    return conc

extended_z_arr = np.linspace(0, 4.85, 1000)
conc_test_1e12 = concentration(extended_z_arr, 1e12 * UNITS.MSun)
conc_test_1e13 = concentration(extended_z_arr, 1e13 * UNITS.MSun)
conc_test_1e14 = concentration(extended_z_arr, 1e14 * UNITS.MSun)
conc_test_1e15 = concentration(extended_z_arr, 1e15 * UNITS.MSun)
```

[9]:
```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 4))
density_conc_vars = np.load('paper_plots_notebook/density_conc_vars.npy')
extended_z_arr = np.linspace(0, 4.85, 1000)
r_array_5e1 = np.geomspace(1e-2, 5e1, 50)

# Subplot 1
ax1.plot(extended_z_arr, conc_test_1e12, label=r'$M_h$ = $10^{12} \,␣
↪M_{\odot}$')
#ax1.plot(extended_z_arr, conc_test_1e13, label=r'$M_h$ = $10^{13} \,␣
↪M_{\odot}$')
#ax1.plot(extended_z_arr, conc_test_1e14, label=r'$M_h$ = $10^{14} \,␣
↪M_{\odot}$')
ax1.plot(extended_z_arr, conc_test_1e15, label=r'$M_h$ = $10^{15} \,␣
↪M_{\odot}$')
ax1.set_xlabel('Redshift', fontsize=14)
ax1.set_xlim(0, 4.85)
ax1.set_ylabel('c(z)', fontsize=14)
ax1.tick_params(axis='y')
ax1.tick_params(axis='y', which='both', direction='in', right=True)
ax1.legend(markerfirst = False, frameon = False, fontsize = 13)


# Subplot 2
ax2.loglog(r_array_5e1, density_conc_vars[0, 0, :], label = 'c(z)', linewidth =␣
↪7)
ax2.loglog(r_array_5e1, density_conc_vars[1, 0, :], label = 'c(z=0) = 9')
ax2.loglog(r_array_5e1, density_conc_vars[2, 0, :], label = 'c = 7')
```

```
ax2.loglog(r_array_5e1, density_conc_vars[3, 0, :], label = 'c = 5')
ax2.loglog(r_array_5e1, density_conc_vars[4, 0, :], label = 'c = 3')
ax2.legend(frameon = False, fontsize = 12)
ax2.set_xlabel('r (Mpc / h)', fontsize=14)
ax2.set_ylabel(r'$n_{\nu} / \overline{n_{\nu}\,}$', fontsize=15)
ax2.set_title(r'$M_h$ = $10^{12} \, M_{\odot}$', fontsize=14)
ax2.set_xlim(1e-2, 3)

#Subplot 3
ax3.loglog(r_array_5e1, density_conc_vars[0, 1, :], label = 'c(z)', linewidth = 7)
ax3.loglog(r_array_5e1, density_conc_vars[1, 1, :], label = 'c = 9')
ax3.loglog(r_array_5e1, density_conc_vars[2, 1, :], label = 'c = 7')
ax3.loglog(r_array_5e1, density_conc_vars[3, 1, :], label = 'c(z=0) = 4.4')
ax3.loglog(r_array_5e1, density_conc_vars[4, 1, :], label = 'c = 3')
ax3.legend(frameon = False, fontsize = 12)
ax3.set_xlabel('r (Mpc / h)', fontsize=14)
ax3.set_ylabel(r'$n_{\nu} / \overline{n_{\nu}\,}$', fontsize=15)
ax3.set_title(r'$M_h$ = $10^{15} \, M_{\odot}$', fontsize=14)
ax3.set_xlim(1e-2, 50)

plt.tight_layout()
#plt.savefig('plot_save_vehicle.pdf')
plt.show()
```



[ ]:

```

```

0.0.5 Enclosed mass for mass conservation

[10]:

```
def compute_enclosed_mass(radii, density_profile, radius_index):
    if radius_index < 0 or radius_index >= len(radii):
        raise ValueError("Radius index out of range")
    # Compute enclosed mass
    enclosed_mass_profile = 0
    for i in range(radius_index + 1):
        if i == 0:
            r1 = 0
```

```python
        else:
            r1 = radii[i - 1]
        r2 = radii[i]
        volume = (4/3) * np.pi * (r2**3 - r1**3)
        enclosed_mass_profile += density_profile[i] * volume

    return enclosed_mass_profile

def compute_enclosed_mass_background(radii, radius_index):
    if radius_index < 0 or radius_index >= len(radii):
        raise ValueError("Radius index out of range")
    # Compute enclosed mass
    enclosed_mass_background = 0
    for i in range(radius_index + 1):
        if i == 0:
            r1 = 0
        else:
            r1 = radii[i - 1]
        r2 = radii[i]

        volume = (4/3) * np.pi * (r2**3 - r1**3)
        enclosed_mass_background += volume  # Density is 1 for the background

    return enclosed_mass_background
```

[11]:

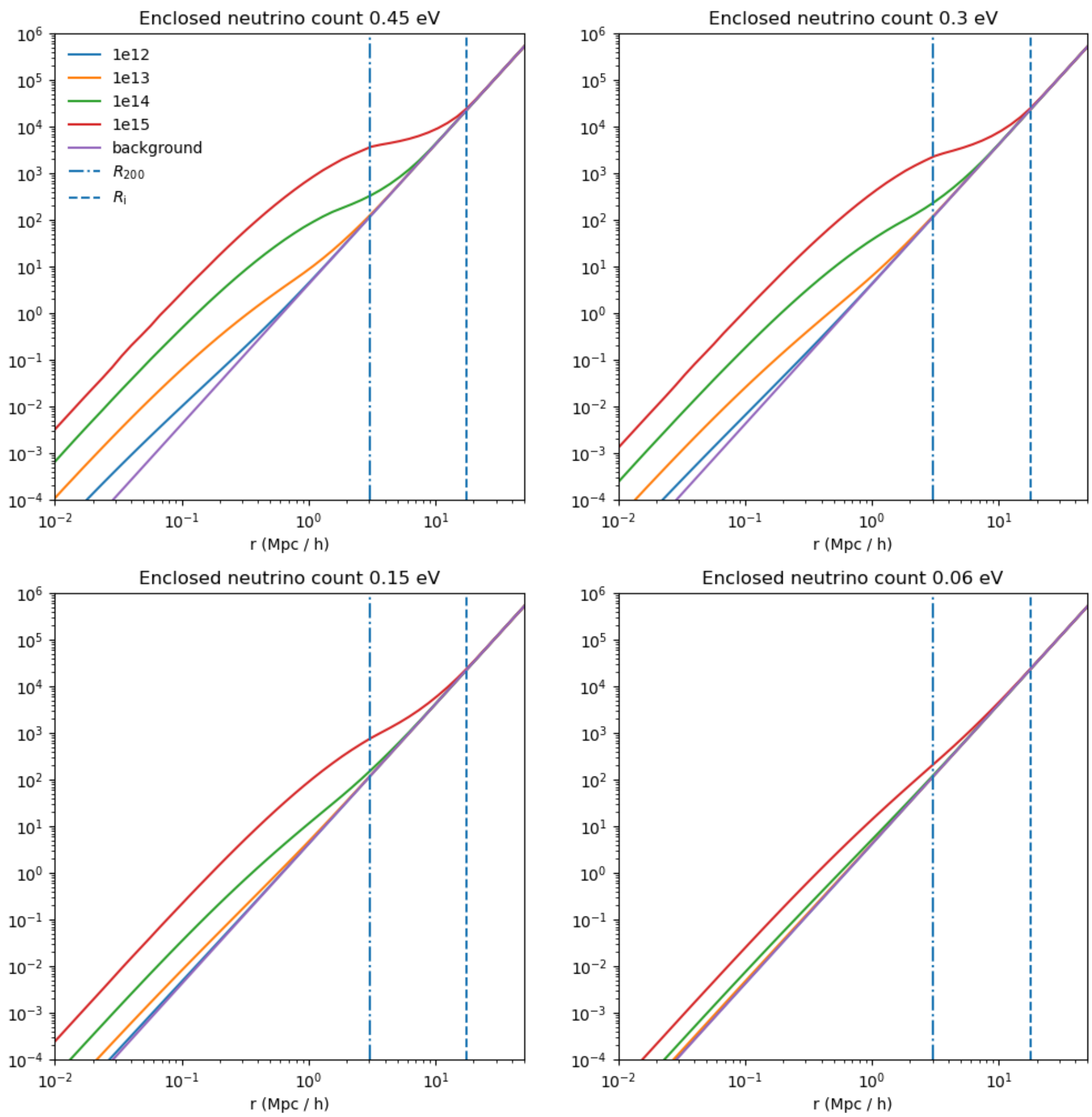
```python
dens_4halos_4nu = np.load('paper_plots_notebook/dens_4halos_4nu.npy')
dm_mass_count = dens_4halos_4nu.shape[0]  # Number of DM halo masses
nu_mass_count = dens_4halos_4nu.shape[2]   # Number of neutrino masses
radii = r_array_5e1.copy()

# Compute the background enclosed mass for all radii
background_encl = []
for radius_index in range(50):
    enclosed_mass_background = compute_enclosed_mass_background(radii,␣
 ↪radius_index)
    background_encl.append(enclosed_mass_background)

# Convert the background enclosed mass list to a numpy array
background_encl = np.array(background_encl)

# Loop over DM and neutrino mass combinations
for dm_idx in range(4):  # 4 different DM mass values
    for nu_idx in range(4):  # 4 different neutrino mass values
        # Extract the density profile for the current combination of DM mass␣
 ↪and neutrino mass
```

```
        density_profile = dens_4halos_4nu[dm_idx, :, nu_idx]  # [50] profile␣
↪for this combination
        mass_conservation_check = []
        profile_encl = []

        for i in range(50):
            radius_index = i

            # Compute the enclosed mass for the profile
            enclosed_mass_profile = compute_enclosed_mass(radii,␣
↪density_profile, radius_index)
            profile_encl.append(enclosed_mass_profile)

            ratio = enclosed_mass_profile / background_encl[i]  # Use the␣
↪precomputed background
            mass_conservation_check.append(ratio)

        # Convert lists to numpy arrays
        profile_encl = np.array(profile_encl)
        mass_conservation_check = np.array(mass_conservation_check)

        # Assign the arrays to dynamically named variables
        suffix = f'_dm{dm_idx}_nu{nu_idx}'
        globals()['profile_encl' + suffix] = profile_encl
        globals()['mass_conservation_check' + suffix] = mass_conservation_check
```

[12]:

```
fig, axes = plt.subplots(2, 2, figsize=(12, 12))

axes[0, 0].loglog(radii, profile_encl_dm0_nu3, label='1e12')
axes[0, 0].loglog(radii, profile_encl_dm1_nu3, label='1e13')
axes[0, 0].loglog(radii, profile_encl_dm2_nu3, label='1e14')
axes[0, 0].loglog(radii, profile_encl_dm3_nu3, label='1e15')
axes[0, 0].loglog(radii, background_encl, label='background')
axes[0, 0].set_xlabel('r (Mpc / h)')
axes[0, 0].set_title('Enclosed neutrino count 0.45 eV')
axes[0, 0].vlines(3.026, 1e-4, 1e6, linestyle='dashdot',␣
↪label=r'$R_{\mathrm{200}}$')
axes[0, 0].vlines(17.7, 1e-4, 1e6, linestyle='dashed',␣
↪label=r'$R_{\mathrm{i}}$')
axes[0, 0].legend(frameon=False, loc='upper left')

axes[0, 1].loglog(radii, profile_encl_dm0_nu2, label='1e12')
axes[0, 1].loglog(radii, profile_encl_dm1_nu2, label='1e13')
axes[0, 1].loglog(radii, profile_encl_dm2_nu2, label='1e14')
axes[0, 1].loglog(radii, profile_encl_dm3_nu2, label='1e15')
axes[0, 1].loglog(radii, background_encl)
axes[0, 1].set_xlabel('r (Mpc / h)')
```

```
axes[0, 1].set_title('Enclosed neutrino count 0.3 eV')
axes[0, 1].vlines(3.026, 1e-4, 1e6, linestyle='dashdot', label=r'$R_{\mathrm{200}}$')
axes[0, 1].vlines(17.7, 1e-4, 1e6, linestyle='dashed', label=r'$R_{\mathrm{i}}$')

axes[1, 0].loglog(radii, profile_encl_dm0_nu1, label='1e12')
axes[1, 0].loglog(radii, profile_encl_dm1_nu1, label='1e13')
axes[1, 0].loglog(radii, profile_encl_dm2_nu1, label='1e14')
axes[1, 0].loglog(radii, profile_encl_dm3_nu1, label='1e15')
axes[1, 0].loglog(radii, background_encl)
axes[1, 0].set_xlabel('r (Mpc / h)')
axes[1, 0].set_title('Enclosed neutrino count 0.15 eV')
axes[1, 0].vlines(3.026, 1e-4, 1e6, linestyle='dashdot', label=r'$R_{\mathrm{200}}$')
axes[1, 0].vlines(17.7, 1e-4, 1e6, linestyle='dashed', label=r'$R_{\mathrm{i}}$')

axes[1, 1].loglog(radii, profile_encl_dm0_nu0, label='1e12')
axes[1, 1].loglog(radii, profile_encl_dm1_nu0, label='1e13')
axes[1, 1].loglog(radii, profile_encl_dm2_nu0, label='1e14')
axes[1, 1].loglog(radii, profile_encl_dm3_nu0, label='1e15')
axes[1, 1].loglog(radii, background_encl)
axes[1, 1].set_xlabel('r (Mpc / h)')
axes[1, 1].set_title('Enclosed neutrino count 0.06 eV')
axes[1, 1].vlines(3.026, 1e-4, 1e6, linestyle='dashdot', label=r'$R_{\mathrm{200}}$')
axes[1, 1].vlines(17.7, 1e-4, 1e6, linestyle='dashed', label=r'$R_{\mathrm{i}}$')

for ax in axes.flatten():
    ax.set_xlim(1e-2, 5e1)
    ax.set_ylim(1e-4, 1e6)

plt.tight_layout
#plt.savefig('plot_save_vehicle.pdf')
plt.show()
```


Enclosed neutrino count 0.45 eV
Enclosed neutrino count 0.3 eV
Enclosed neutrino count 0.15 eV
Enclosed neutrino count 0.06 eV
1e12
1e13
1e14
1e15
background
R_{200}
R_{i}
r (Mpc / h)

[ ]: