# Cluster_WL Documentation

*Release 1.0*

**Tom McClintock**

**Nov 21, 2019**

# Getting Started

Cluster Toolkit is a Python package specifically built for calculating weak lensing signals from galaxy clusters and cluster cosmology. It consists of a Python front end wrapped around a well optimized back end in C, merged with cffi. The core functionality of the package includes:

- 3D density functions $\rho(r)$
- 3D correlation functions $\xi(r)$
- Halo bias models $b(M)$
- Projected density and differential profiles $\Sigma(R)$ and $\Delta\Sigma$
- Radially averaged profiles $\overline{\Delta\Sigma}$
- Boost factor models $\mathcal{B} = (1 - f_{\rm cl})^{-1}$
- Miscentering effects on projected profiles $R_{\rm mis}$
- Halo mass functions $\frac{dn}{dM}(M, z)$
- Mass-concentration relations $M - c$
- Sunyaev-Zel'dovich (SZ) cluster signals $Y_{SZ}$ (in development)
- Cluster magnification $\kappa(\theta)$ and shear profiles $\gamma(\theta)$ (in development)

The source code is publically available at https://github.com/tmcclintock/cluster_toolkit.

---

**Note:** Unless stated otherwise, all distances are assumed to be $\mathrm{Mpc}/h$ comoving and masses $\mathrm{M}_\odot/h$. Furthermore, power spectra $P(k)$ must be in units of $(\mathrm{Mpc}/h)^3$ with wavenumber $k$ in units of $h/\mathrm{Mpc}$.

---

CHAPTER 1

# Installation

To install the cluster_toolkit you currently need to build it from source:

```
git clone https://github.com/tmcclintock/cluster_toolkit.git
cd cluster_toolkit
python setup.py install
```

To run the tests you can do:

```
python setup.py test
```

## 1.1 Requirements

This package has only ever been tested with Python 2.7.x and has some dependencies. The Python dependencies that you can get with pip are:

- Numpy: 1.13 or later
- cffi: 1.10 or later
- pytest: 3.x or later for testing

In addition, you must have the GNU Science Library (GSL) installed. If you follow the instructions in their INSTALL file you will be done in only a few lines in the terminal. There is a pip installable GSL, but I do not know if it will work with the cluster toolkit.

Furthermore, while it is not a dependency of this code, for some of the functions you will need a way to calculate the linear and nonlinear matter power spectra $P(k)$. Two good options are CAMB and CLASS. Both are also available in the Core Cosmology Library.

CHAPTER 2

# Density Profiles

Most cluster observables are closely related to their density profiles. This repository contains routines for calcalating the NFW and Einasto profiles.

## 2.1 NFW Profile

The NFW profile (arxiv) is a 3D density profile given by:

$$\rho_{\rm nfw}(r) = \frac{\Omega_m \rho_{\rm crit} \delta_c}{\left(\frac{r}{r_s}\right)\left(1+\frac{r}{r_s}\right)^2}.$$

The free parameters are the cluster mass $M_\Delta$ and concentration $c_\Delta = r_\Delta / r_s$. In this module we choose to define the density with respect to the matter background density $\Omega_m \rho_{\rm crit}$. The scale radius $r_s$ is given in $h^{-1}{\rm Mpc}$, however the code uses concentration $c_\Delta$ as an argument instead. The normalization $\delta_c$ is calculated internally and depends only on the concentration. As written, because of the choice of units the only cosmological parameter that needs to be passed in is $\Omega_m$.

**Note:** The density profiles can use $\Delta \neq 200$.

To use this, you would do:

```
from cluster_toolkit import density
import numpy as np
radii = np.logspace(-2, 3, 100) #Mpc/h comoving
mass = 1e14 #Msun/h
concentration = 5 #arbitrary
Omega_m = 0.3
rho_nfw = density.rho_nfw_at_r(radii, mass, concentration, Omega_m)
```

## 2.2 Einasto Profile

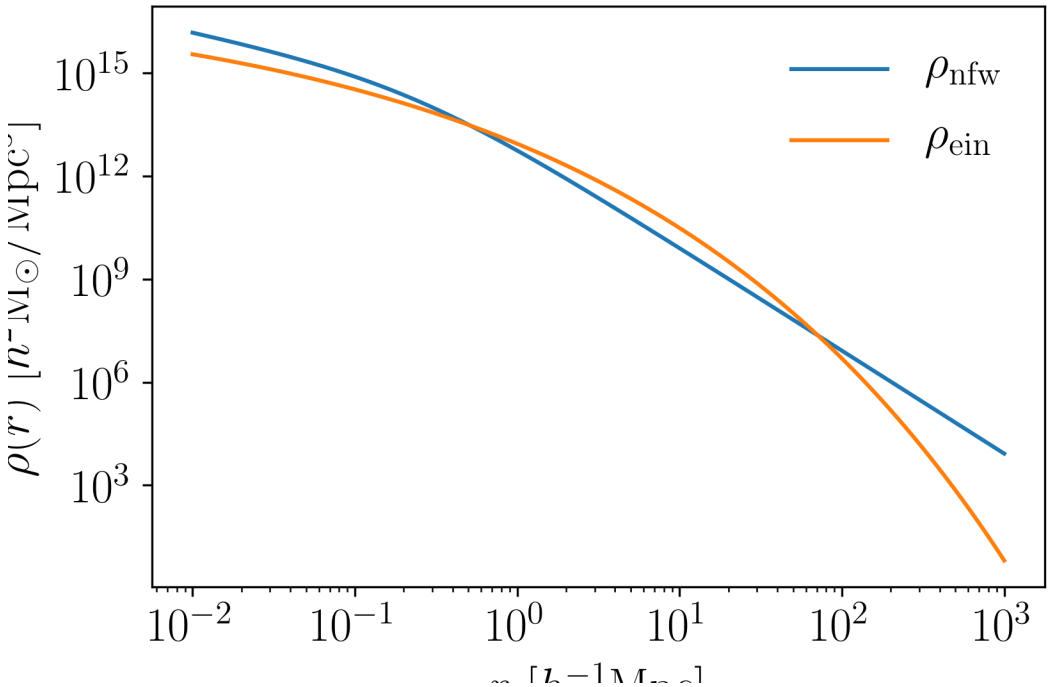
The Einasto profile is a 3D density profile given by:

$$\rho_{\rm ein}(r) = \rho_s \exp\left(-\frac{2}{\alpha}\left(\frac{r}{r_s}\right)^{\alpha}\right)$$

In this model, the free parameters are the scale radius $r_s$, $\alpha$, and the cluster mass $M_\Delta$. The scale density $\rho_s$ is calculated internally, or can be passed in instead of mass. To use this, you would do:

```
from cluster_toolkit import density
import numpy as np
radii = np.logspace(-2, 3, 100) #Mpc/h comoving
mass = 1e14 #Msun/h
r_scale = 1.0 #Mpc/h comoving scale radius
alpha = 0.19 #arbitrary; a typical value
Omega_m = 0.3
rho_ein = density.rho_einasto_at_r(radii, mass, r_scale, alpha, Omega_m)
```

We can see the difference between these two profiles here:



# CHAPTER 3

# Correlation Functions

Cluster density profiles are closely related to the correlation function

$$\rho(r) = \rho_0(1 + \xi_{\rm hm}(r))$$

That is, the average density of halos some distance $r$ from the center of the halo is proportional to mean density $\rho_0 = \Omega_m \rho_{\rm crit}$ of the universe and the *halo-matter* correlation function, or the tendency to find matter near halos. This module makes various correlation functions available. The correlation functions for the NFW and Einasto profiles can also be computed directly from the density profiles by inverting the above equation.

**Note:** By definition $\int {\rm d}V\ \xi(\vec{r}) = 0$. Almost no analytic correlation function has this built in, however.

## 3.1 NFW Profile

The NFW profile (arxiv) is a 3D correaltion function given by:

$$\xi_{\rm nfw}(r) = \frac{\rho_{\rm nfw}(r)}{\Omega_m \rho_{\rm crit}} - 1$$

The free parameters are the cluster mass $M_\Delta$ and concentration $c_\Delta = r_\Delta / r_s$. In this module we choose to define the density with respect to the matter background density $\Omega_m \rho_{\rm crit}$. The scale radius $r_s$ is given in $h^{-1}{\rm Mpc}$, however the code uses concentration $c_\Delta$ as an argument instead. As written, because of the choice of units the only cosmological parameter that needs to be passed in is $\Omega_m$. The arguments are identical to the density profile.

**Note:** The correlation functions can use $\Delta \neq 200$ as an argument: `delta=200`.

To use this, you would do:

```
from cluster_toolkit import xi
import numpy as np
radii = np.logspace(-2, 3, 100) #Mpc/h comoving
mass = 1e14 #Msun/h
concentration = 5 #arbitrary
Omega_m = 0.3
xi_nfw = xi.xi_nfw_at_r(radii, mass, concentration, Omega_m)
```

## 3.2 Einasto Profile

The Einasto profile is a 3D density profile given by:

$$\xi_{\rm ein}(r) = \frac{\rho_{\rm ein}}{\Omega_m \rho_{\rm crit}} - 1$$

In this model, the free parameters are the scale radius $r_s$, $\alpha$, and the cluster mass $M_\Delta$. The scale density $\rho_s$ is calculated internally, or can be passed in instead of mass. This is the same arguments as the density profile. To use this, you would do:

```
from cluster_toolkit import xi
import numpy as np
radii = np.logspace(-2, 3, 100) #Mpc/h comoving
mass = 1e14 #Msun/h
r_scale = 1.0 #Mpc/h comoving scale radius
alpha = 0.19 #arbitrary; a typical value
Omega_m = 0.3
xi_ein = xi.xi_einasto_at_r(radii, mass, r_scale, alpha , Omega_m)
```

## 3.3 1-halo vs. 2-halo Terms

The NFW and Einasto profiles describe the "1-halo" density of halos, or the density of a halo within its boundary. However, halos tend to be found near other halos, and for this reason the *average density* of halos should have a 2-halo term. In other words, as you move far from the center of a halo (e.g. past the scale radius or $r_{200}$ or a splashback radius or any arbitrary boundary) the halo-matter correlation function becomes the "2-halo" term, or the likelihood that one is in a second halo

$$\xi_{\rm hm}(r >> r_s) = \xi_{\rm 2-halo}(r).$$

The two halo term, as detailed below, should be related to the matter density field and a bias. The specific treatment of this is not unanimously agreed upon.

## 3.4 Matter Correlation Function

The matter (auto)correlation function describes the average density of matter.

$$\rho_{\rm m}(r) = \rho_0(1 + \xi_{\rm mm}(r))$$

By definition it is related to the matter power spectrum by a Fourier transform

$$\xi_{\rm mm}(r) = \frac{1}{2\pi^2}\int_0^\infty {\rm d}k\; k^2 P(k) \frac{\sin kr}{kr}.$$

There is not consensus on what power spectrum to use. Hayashi & White use the *linear* matter power spectrum, while Zu et al. use the *nonlinear* matter power spectrum. Anecdotally, the former is generally better for lower mass halos and the latter is better for higher mass halos. Regardless of what you use, to call this you would do

```
from cluster_toolkit import xi
import numpy as np
radii = np.logspace(-2, 3, 100) #Mpc/h comoving
#Assume that k and P come from somewhere, e.g. CAMB or CLASS
xi_mm = xi.xi_mm_at_r(radii, k, P)
```

## 3.5 2-halo Correlation Function

Halos are *biased* tracers of the matter density field, meaning the 2-halo correlation function is

$$\xi_{2-\mathrm{halo}}(r, M) = b(M)\xi_{\mathrm{mm}}(r)$$

The bias is described in more detail in the bias section of this documentation (in progress). To calculate the 2-halo term you would do

```
from cluster_toolkit import xi
from cluster_toolkit import bias
import numpy as np
radii = np.logspace(-2, 3, 100) #Mpc/h comoving
mass = 1e14 #Msun/h
Omega_m = 0.3
#Assume that k and P come from somewhere, e.g. CAMB or CLASS
xi_mm = xi.xi_mm_at_r(radii, k, P)
#Assume that k and P_linear came from somewhere, e.g. CAMB or CLASS
bias = bias.bias_at_M(mass, k, P_linear, Omega_m)
xi_2halo = xi.xi_2halo(bias, xi_mm)
```

## 3.6 Halo-matter Correlation Function

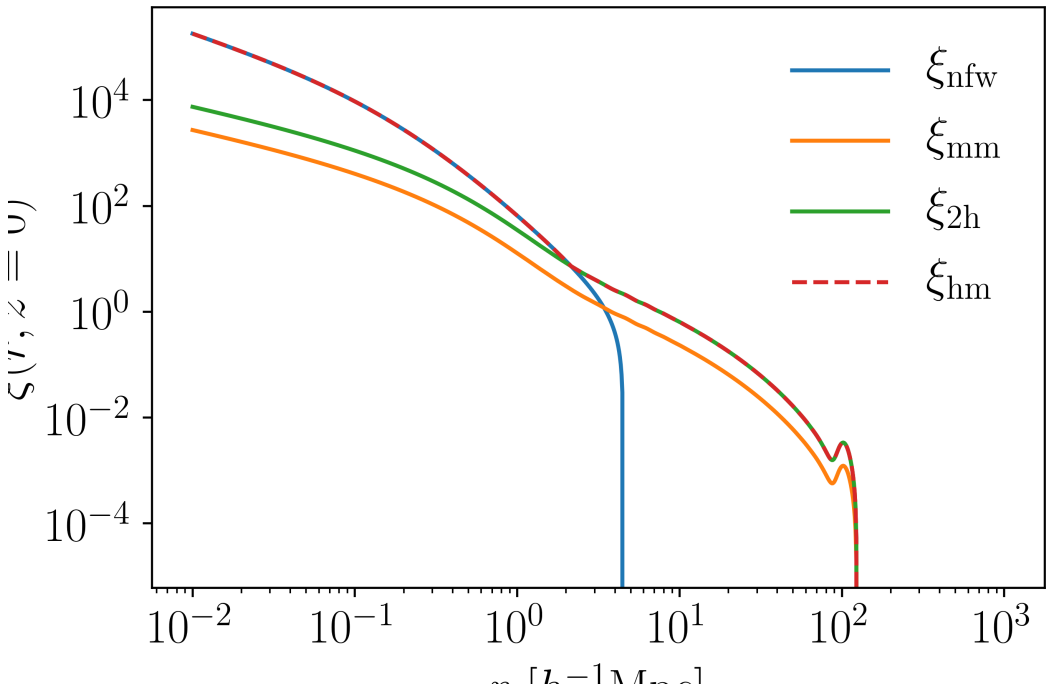
At small scales, the correlation function follows the 1-halo term (e.g. NFW or Einasto) while at large scales it follows the 2-halo term. There is no consensus on how to combine the two. Zu et al. take the max of the two terms, while Chang et al. sum the two. The default behavior of this module is to follow Zu et al., and in the near future it will be easy to switch between different options. Mathematically this is

$$\xi_{\mathrm{hm}}(r, M) = \max(\xi_{1-\mathrm{halo}}, \xi_{2-\mathrm{halo}}).$$

To use this you would do

```
from cluster_toolkit import xi
#Calculate 1-halo and 2-halo terms here
xi_hm = xi.xi_hm(xi_1halo, xi_2halo)
```

Here are each of these correlation functions plotted together:


$\xi_{\rm nfw}$
$\xi_{\rm mm}$
$\xi_{\rm 2h}$
$\xi_{\rm hm}$
$10^4$
$10^2$
$10^0$
$10^{-2}$
$10^{-4}$
$10^{-2}$
$10^{-1}$
$10^0$
$10^1$
$10^2$
$10^3$

# CHAPTER 4

# Halo Bias

Halos, which host galaxies and galaxy clusters, are *biased* tracers of the matter density field. This means at large scales the correlation function is

$$\xi_{\rm hm}(R) = b\xi_{\rm mm}$$

where the bias is a function of mass (and cosmological parameters). This module implements the Tinker et al. 2010 halo bias model, which is accurate to 6%. Other biases will be available in the future. To use this you would do:

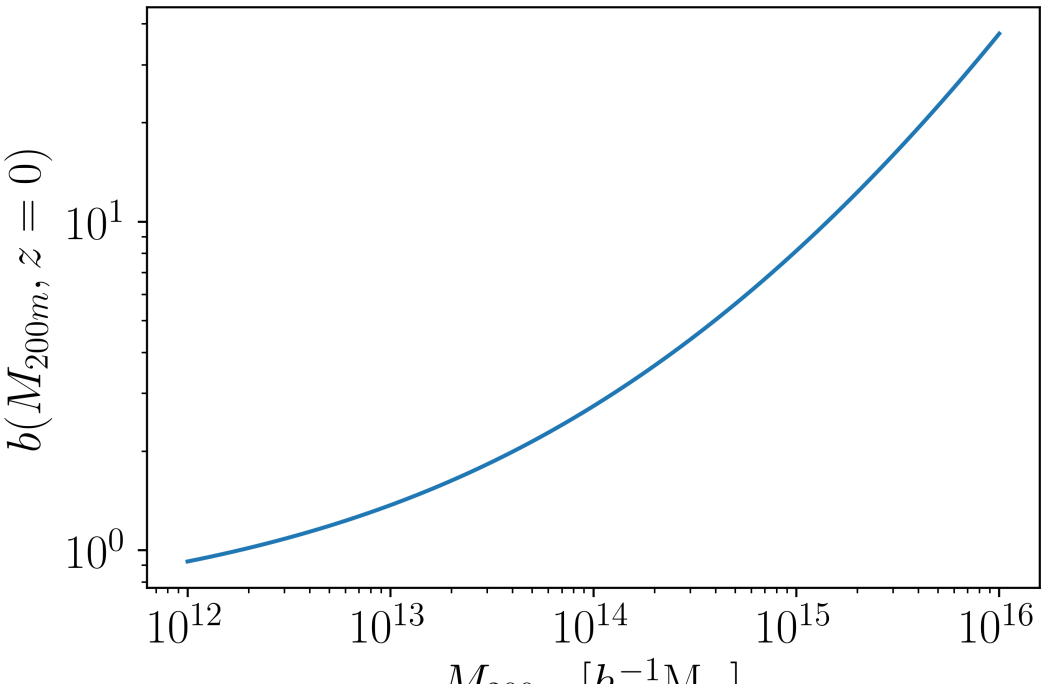
```
from cluster_toolkit import bias
mass = 1e14 #Msun/h
Omega_m = 0.3
#Assume that k and P_linear came from somewhere, e.g. CAMB or CLASS
bias = bias.bias_at_M(mass, k, P_linear, Omega_m)
```

**Note:** The bias can use $\Delta \neq 200$ as an argument `delta=200`.

This module also allows for conversions between mass and RMS density variance $\sigma^2$ and peak height $\nu$.

```
from cluster_toolkit import bias
mass = 1e14 #Msun/h
Omega_m = 0.3
#Assume that k and P_linear came from somewhere, e.g. CAMB or CLASS
sigma2= bias.sigma2_at_M(mass, k, P_linear, Omega_m)
nu = bias.nu_at_M(Mass, k, P_linear, Omega_m)
```

The bias as a function of mass is seen here for a basic cosmology:


$b(M_{200m}, z = 0)$
$10^1$
$10^0$
$10^{12}$
$10^{13}$
$10^{14}$
$10^{15}$
$10^{16}$

# CHAPTER 5

# Projected Density profiles $\Sigma$ and $\Delta\Sigma$

Weak lensing measurements of galaxy clusters involve calculating the projected and differential density profiles of the cluster.

## 5.1 Surface Mass Density $\Sigma(R)$

The projected density (or the surface mass density) is defined as

$$\Sigma(R) = \Omega_m \rho_{\rm crit} \int_{-\infty}^{+\infty} {\rm d}z \; \xi_{\rm hm}(\sqrt{R^2 + z^2}).$$

Where $\xi_{hm}$ is the halo-matter correlation function (link to the correlation function documentation). The integral is along the line of site, meaning that $R$ is the distance on the sky from the center of the cluster.

**Note:** $\Sigma$ and $\Delta\Sigma$ use units of $h{\rm M}_\odot/{\rm pc}^2$, following convention in the literature.

**Note:** This module is called `cluster_toolkit.deltasigma`, even though it contains routines to calculate $\Sigma$ as well as $\Delta\Sigma$.

To calculate this using the module you would use:

```
from cluster_toolkit import deltasigma
mass = 1e14 #Msun/h
concentration = 5 #arbitrary
Omega_m = 0.3
R_perp = np.logspace(-2, 2.4, 100) #Mpc/h comoving; distance on the sky
#Assume that radii and xi_hm are computed here
Sigma = deltasigma.Sigma_at_R(R_perp, radii, xi_hm, mass, concentration, Omega_m)
```

## 5.2 NFW $\Sigma(R)$

The example code above computes a $\Sigma$ given any halo-matter correlation function, but you can compute $\Sigma_{nfw}$ directly using

```
from cluster_toolkit import deltasigma
mass = 1e14 #Msun/h
concentration = 5 #arbitrary
Omega_m = 0.3
R_perp = np.logspace(-2, 2.4, 100) #Mpc/h comoving; distance on the sky
Sigma_nfw = deltasigma.Sigma_nfw_at_R(R_perp, mass, concentration, Omega_m)
```

If you know of an analytic form of the the Einasto profile, please let me know.

## 5.3 Differential Surface Density $\Delta\Sigma(R)$

The differential (or excess) surface mass density is defined as

$$\Delta\Sigma = \bar{\Sigma}(< R) - \Sigma(R)$$

where $\Sigma$ is given above and

$$\bar{\Sigma}(< R) = \frac{2}{R^2}\int_0^R \mathrm{d}R'\ R'\Sigma(R'),$$

or the average surface mass density within the circle of radius:math:*R*. To calculate this you would use

```
from cluster_toolkit import deltasigma
mass = 1e14 #Msun/h
concentration = 5 #arbitrary
Omega_m = 0.3
#Assume that Sigma at Rp is calculated here
R_perp = np.logspace(-2, 2.4, 100) #Mpc/h comoving; distance on the sky
DeltaSigma = deltasigma.DeltaSigma_at_R(R_perp, Rp, Sigma, Mass, concentartion, Omega_
↪m)
```

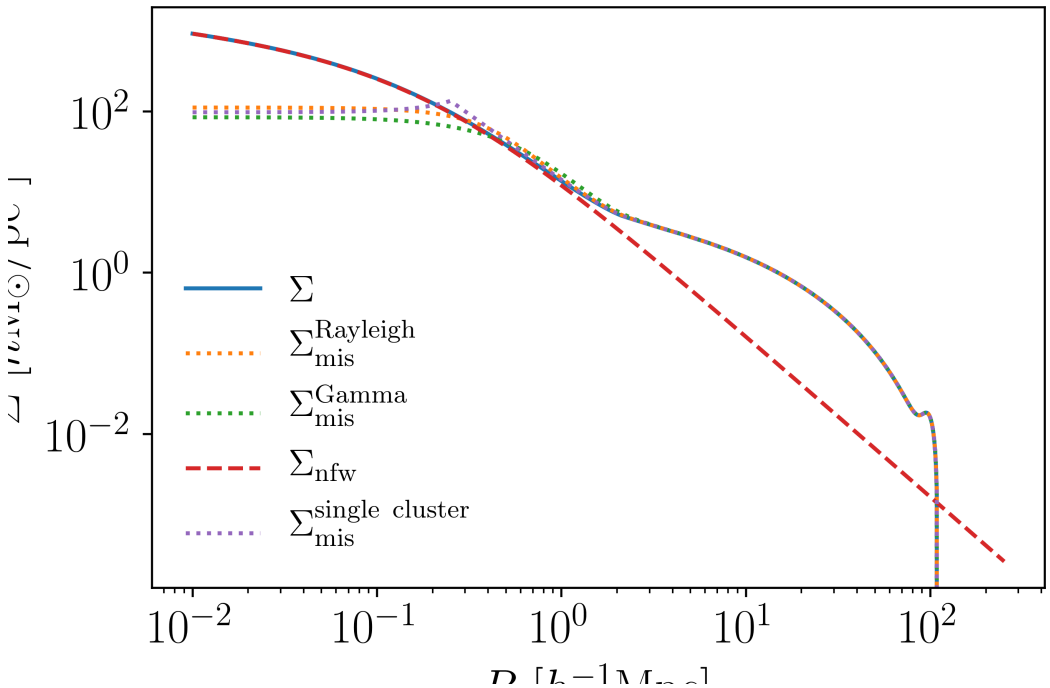
As you can see, the code is structured so that the input $\Sigma$ profile is arbitrary.

---

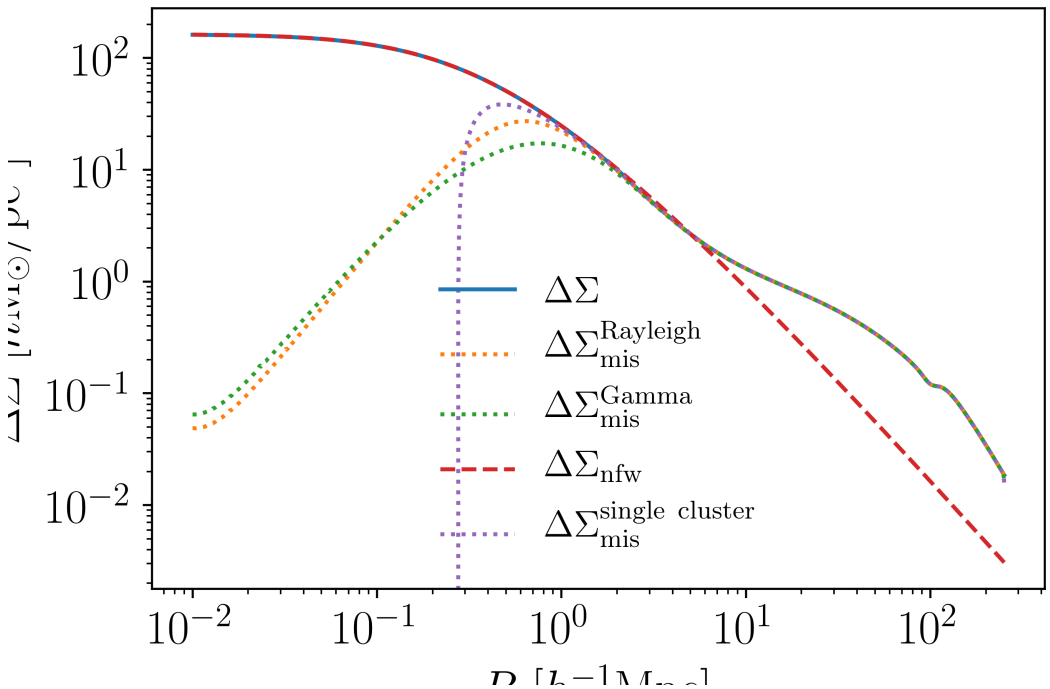
**Note:** Mass, concentration, and $\Omega_m$ are also arguments to $\Delta\Sigma$, because an NFW profile is used to extrapolate the integrand for $\bar{\Sigma}(< R)$ at very small scales. To avoid issues when using an Einasto or other profile, make sure that the input profiles are calculated to fairly large and small scales.

---

This figure shows the different $\Sigma(R)$ profiles, including with miscentering




This figure shows the different $\Delta\Sigma(R)$ profiles, including with miscentering



# CHAPTER 6

## Radially Averaged Projected Profiles

Weak lensing measurements are bin-averaged quantities. That is, they are measurements of a quantity with a radial bin around the lens. This module allows for calculating radially averaged projected profiles from continuous profiles. Mathematically this is

$$\overline{\Delta\Sigma} = \frac{2}{R_2^2 - R_1^2} \int_{R_1}^{R_2} \mathrm{d}R' R' \Delta\Sigma(R').$$

This can be computed in the code by using

```
from cluster_toolkit import averaging
#Assume DeltaSigma at R_perp are computed here
N_bins = 15
bin_edges = np.logspace(np.log10(0.2), np.log10(30.), N_bins+1)
#Bin edges are from 200 kpc/h to 30 Mpc/h
averaged_DeltaSigma = averaging.average_profile_in_bins(bin_edges, R_perp, DeltaSigma)
```

**Note:** The `average_profile_in_bins` function can work with any projected profile.

**Note:** The returned average profile will be an array of length $N_{\mathrm{bins}}$.

# CHAPTER 7

# Boost Factors

In galaxy cluster weak lensing, a significant systematic issue is cluster member dilution also known as boost factors. The idea is that if some cluster galaxies are misidentified as source (background) galaxies, then your weak lensing signal is diluted due to the fact that the cluster member galaxy won't be sheared or magnified. Traditionally, one calculates or estimates this correction and "boosts" the data vector. This boost factor is radially dependent, since you will tend to misidentify cluster members close to the cluster more than those farther out. Mathematically this looks like

$$\Delta\Sigma_{\rm corrected}(R) = (1 - f_{\rm cl})^{-1}(R)\Delta\Sigma(R)$$

where $f_{\rm cl}$ is the fraction of cluster members misidentified as being source galaxies. For shorthand, we write $\mathcal{B} = (1 - f_{\rm cl})^{-1}$. This module provides multiple models for $\mathcal{B}$.

## 7.1 NFW Boost Model

In McClintock et al. (in prep.) we model the boost factor with an NFW model:

$$\mathcal{B}(R) = 1 + B_0 \frac{1 - F(x)}{x^2 - 1}$$

where $x = R/R_s$ and

$$F(x) = \begin{cases} to \\ \\ \dfrac{\tan^{-1}\sqrt{x^2-1}}{\sqrt{x^2-1}} : x > 1 \\ \\ 1 : x = 1 \\ \\ \dfrac{\tanh^{-1}\sqrt{1-x^2}}{\sqrt{1-x^2}} : x < 1. \end{cases} \tag{7.1}$$

Parameters that need to be specified by the user are $B_0$ and the scale radius $R_s$. To use this, you would do:

```
from cluster_toolkit import boostfactors
import numpy as np
R = np.logspace(-2, 3, 100) #Mpc/h comoving
B0 = 0.1 #Typical value
Rs = 1.0 #Mpc/h comoving; typical value
B = boostfactors.boost_nfw_at_R(R, B0, Rs)
```

## 7.2 Powerlaw Boost Model

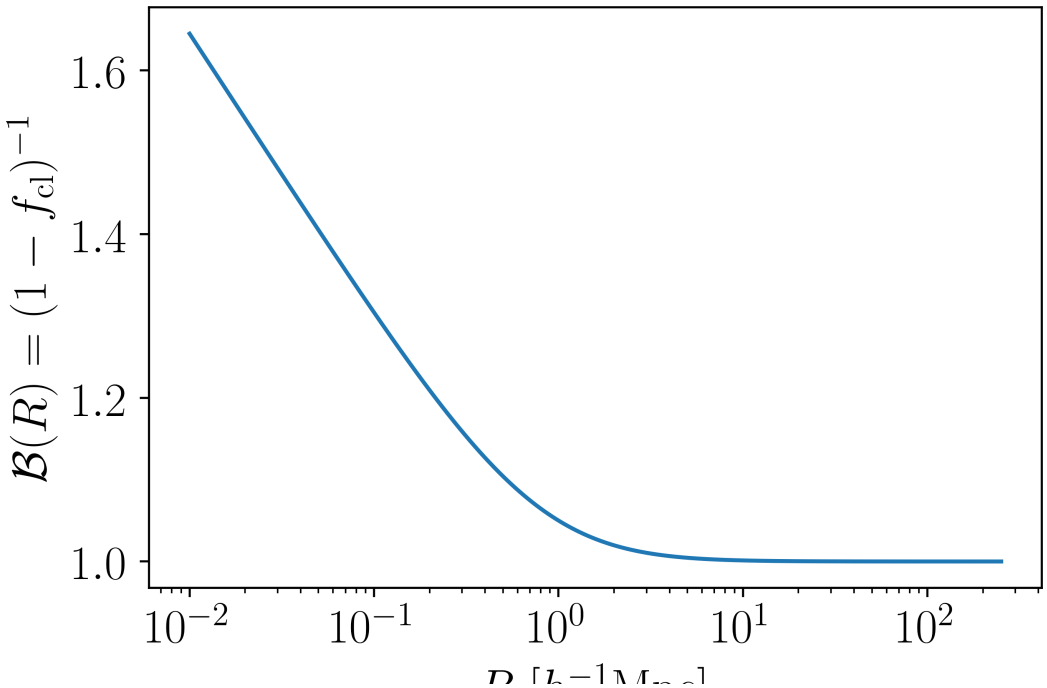
In Melchior et al. we used a power law for the boost factor.

$$\mathcal{B} = 1 + B_0 \left(\frac{R}{R_s}\right)^{\alpha}$$

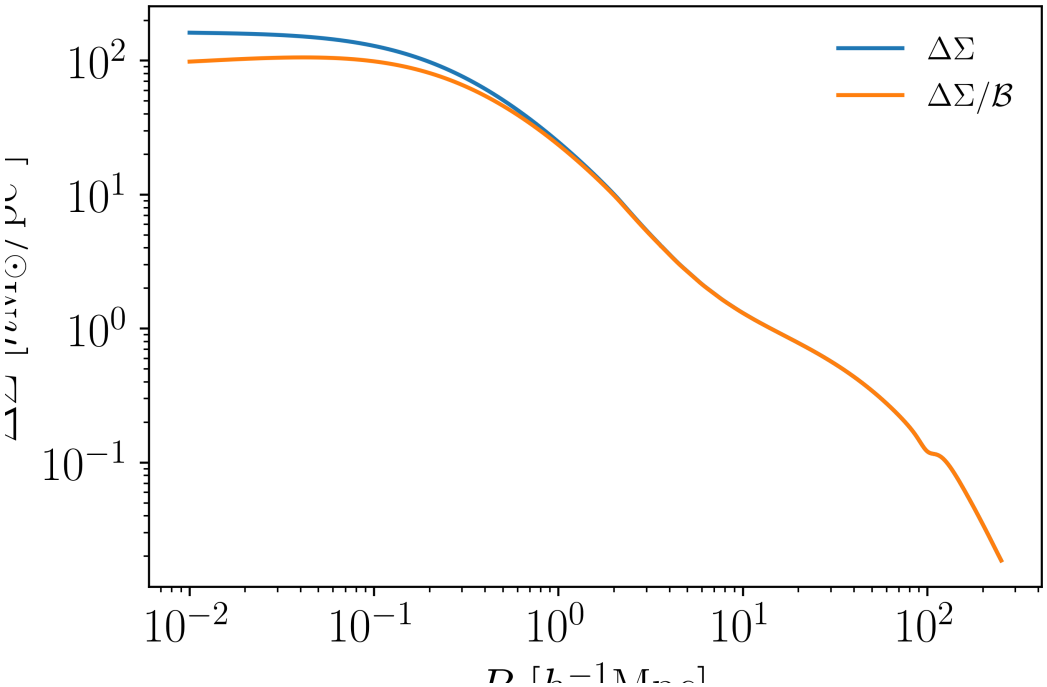
Here, the input parameters are $B_0$, the scale radius $R_s$, and the exponent $\alpha$. This is also available in this module:

```
from cluster_toolkit import boostfactors
import numpy as np
R = np.logspace(-2, 3, 100) #Mpc/h comoving
B0 = 0.1 #Typical value
Rs = 1.0 #Mpc/h comoving; typical value
alpha = -1.0 #arbitrary
B = boostfactors.boost_powerlaw_at_R(R, B0, Rs, alpha)
```

This figure shows the NFW boost factor model:




This figure shows how the boost factor changes the $\Delta\Sigma(R)$ profile:



# CHAPTER 8

# Miscentering Effects

If galaxy cluster centers are not properly identified on the sky, then quantities measured in annuli around that center will not match theoretical models. This effect is detailed in Johnston et al. (2007) and Yang et al. (2006).

To summarize, if a cluster center is incorrectly identified on the sky by a distance $R_{\rm mis}$ then the surface mass density becomes:

$$\Sigma_{\rm mis}^{\rm single\ cluster}(R, R_{\rm mis}) = \int_0^{2\pi} \frac{{\rm d}\theta}{2\pi} \Sigma\left(\sqrt{R^2 + R_{\rm mis}^2 + 2RR_{\rm mis}\cos\theta}\right).$$

That is, the average surface mass density at distance $R$ away from the incorrect center is the average of the circle drawn around that incorrect center. To get the miscentered profiles of a *single cluster* you would use

```
from cluster_toolkit import miscentering
mass = 1e14 #Msun/h
conc = 5 #arbitrary
Omega_m = 0.3
#Calculate Rp and Sigma here, where Sigma is centered
Rmis = 0.25 #Mpc/h; typical value
Sigma_mis_single = miscentering.Sigma_mis_single_at_R(Rp, Rp, Sigma, mass, conc,␣
↪Omega_m, Rmis)
```

As you can see `Rp` is passed in twice. It is first used as the location at which to evaluate `Sigma_mis` and then as the locations at which `Sigma` is known. So if you wanted those two radial arrays can be different.

The $\Delta\Sigma$ profile is defined the usual way

$$\Delta\Sigma(R, R_{\rm mis}) = \bar{\Sigma}_{\rm mis}(< R, R_{\rm mis}) - \Sigma_{\rm mis}(R, R_{\rm mis})$$

which can be calculated using this module using

```
DeltaSigma_mis_single = miscentering.DeltaSigma_mis_at_R(Rp, Rp, Sigma_mis_single)
```

## 8.1 Stacked Miscentering

In a stack of clusters, the amount of miscentering will follow a distribution $P(R'|R_{\rm mis})$ given some characteristic miscentering length $R_{\rm mis}$. That is, some clusters will be miscentered more than others. Simet et al. (2017) for SDSS and Melchior et al. (2017) assume a Raleigh distribution for the amount of miscentering $R'$:

$$P(R'|R_{\rm mis}) = \frac{R'}{R_{\rm mis}^2} \exp[-R'^2/2R_{\rm mis}^2] \,.$$

In McClintock et al. (2019) we used a Gamma profile for the mistnering:

$$P(R'|R_{\rm mis}) = \frac{R'}{R_{\rm mis}^2} \exp[-R'/R_{\rm mis}] \,.$$

Both of these are available in the toolkit. We see that $R_{\rm mis}$ is a free parameter, giving rise to a miscentered projected stacked density profile:

$$\Sigma_{\rm mis}^{\rm stack}(R) = \int_0^\infty {\rm d}R' \; P(R'|R_{\rm mis}) \Sigma_{\rm mis}^{\rm single\ cluster}(R, R')$$

which can then itself be integrated to get $\Delta\Sigma_{\rm mis^{stack}}$. To calculate these in the code you would use:

```
from cluster_toolkit import miscentering
#Assume Sigma at R_perp are computed here
Sigma_mis = miscentering.Sigma_mis_at_R(R_perp, R_perp, Sigma, mass, concentration,␣
↪Omega_m, R_mis)
DeltaSigma_mis = miscentering.DeltaSigma_mis_at_R(R_perp, R_perp, Sigma_mis)
```

# CHAPTER 9

# Halo Mass Function

Clusters and galaxies live inside of dark matter halos, which represent peaks in the matter density field. This means that it is possible to use the abundance of clusters to constrain cosmology if you have well understood mapping from clusters onto halos (which is outside the scope of this code).

The abundance of halos is also known as the *halo mass function*. A general mathematical form for the mass function is given in Tinker et al. (2008) (where they cite Press & Schechter (1974), Jenkins et al. (2000) and other papers that you can look up) is:

$$\frac{\mathrm{d}n}{\mathrm{d}M} = f(\sigma)\frac{\rho_m}{M}\frac{\mathrm{d}\ln\sigma^{-1}}{\mathrm{d}M}.$$

Where $\sigma$ is the RMS variance of a spherical top hat containing a mass $M$, $\rho_m = \rho_{\rm crit}\Omega_m$ is the mean matter density and $f(\sigma)$ is known as the *halo multiplicity function*. Practically speaking, what sets one mass function model apart from another is how the multiplicity is written down.

At some point in the future the toolkit will have other options for the mass function, but for now it implements the mass function from Tinker et al. (2008). Specifically, the version in Appendix C of that paper, which is usually the one people mean when they refer to this paper.

---

**Note:** Implicitely, the mass function depends on the linear matter power spectrum $P(k)$ in order to map from $M$ to $\sigma$. Since the toolkit doesn't have its own power spectrum implemented, the user must input one from, e.g. CLASS or CAMB.

---

## 9.1 Tinker Mass Function

The Tinker mass function is defined by it's multiplicity function, which looks like

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

In this mass function $d$, $e$, $f$, and $g$ are free parameters that depend on halo definition. By default they are set to the values associated with $M_{200m}$. If you want to switch to other values for other halo definitions, see Tinker et al. (2008). The normalization $B$ is calculated internally so that you don't have to pass it in.

To use this in the code you would do:

```
from cluster_toolkit import massfunction
import numpy as np
#Assume that k and P come from somewhere, e.g. CAMB or CLASS
#Units of k and P are h/Mpc and (Mpc/h)^3
Mass = 1e14 #Msun/h
Omega_m = 0.3 #example value
dndM = massfunction.dndM_at_M(Mass, k, P, Omega_m)
#Or could also use an array
Masses = np.logspace(12, 16)
dndM = massfunction.dndM_at_M(Masses, k, P, Omega_m)
```

## 9.2 Binned Mass Functions

In reality in a simulation or in cluster abundance the real observable is the number density of objects in some mass bin of finite width. Written out, this is

$$n = \int_{M_1}^{M_2} \mathrm{d}M \, \frac{\mathrm{d}n}{\mathrm{d}M}.$$

In the toolkit this is available by first calculating $\mathrm{d}n/\mathrm{d}M$ and then passing that back to the toolkit. This is available in the code by using

```
from cluster_toolkit import massfunction
import numpy as np
#Assume that k and P come from somewhere, e.g. CAMB or CLASS
#Units of k and P are h/Mpc and (Mpc/h)^3
Omega_m = 0.3 #example value
M = np.logspace(12, 16) #Msun/h
dndM = massfunction.dndM_at_M(M, k, P, Omega_m)
M1 = 1e12 #Msun/h
M2 = 1e13 #Msun/h
n = n_in_bin(M1, M2, M, dndM)
```

You can also pass in many bin edges at once:

```
edges = np.array([1e12, 5e12, 1e13, 5e13])
n = n_in_bins(edges, M, dndM)
```

# CHAPTER 10

## Mass-concentration Relations

The inner regions of clusters are called the 1-halo regime. The 1-halo regime is often modeled analytically using either an NFW or Einasto profile. Both of these profiles are functions of at least two variables. Numerous papers have examined the relationship between mass and NFW halo concentration. This module implements the Diemer-Kravtzov 2015 M-c relation. At present, only $M_{200c}$ and $M_{200m}$ mass definitions are supported. In the near future this will be expanded to $\Delta \neq 200$ as well as $M_{\rm vir}$.

To call this function you would do the following:

```
from cluster toolkit import concentration as conc
M = 1e14 #Msun/h
Omega_m = 0.3 #Matter fraction
Omega_b = 0.05 #Baryon fraction
ns = 0.96 #Power spectrum index
h = 0.7 #Hubble constant
#Assume that k and P come from somewhere, e.g. CAMB or CLASS
#k are wavenumbers in h/Mpc and P is the linear power spectrum
#in (Mpc/h)^3
#The Mass_type argument can either be 'mean' or 'crit'
Mass_type = 'mean'
c = conc.concentration_at_M(M, k, P, ns, Omega_b, Omega_m, h, Mass_type)
```

CHAPTER 11

cluster_toolkit

## 11.1 cluster_toolkit package

### 11.1.1 Submodules

#### cluster_toolkit.averaging module

Averaging projected cluster profiles.

cluster_toolkit.averaging.**average_profile_in_bin**(*Rlow*, *Rhigh*, *R*, *prof*)
Average profile in a bin.

Calculates the average of some projected profile in a radial bin in Mpc/h comoving.

**Parameters**

- **Rlow** (*float*) – Inner radii.
- **Rhigh** (*float*) – Outer radii.
- **R** (*array like*) – Radii of the profile.
- **prof** (*array like*) – Projected profile.

**Returns** Average profile in the radial bin, or annulus.

**Return type** float

cluster_toolkit.averaging.**average_profile_in_bins**(*Redges*, *R*, *prof*)
Average profile in bins.

Calculates the average of some projected profile in a radial bins in Mpc/h comoving.

**Parameters**

- **Redges** (*array like*) – Array of radial bin edges.
- **R** (*array like*) – Radii of the profile.

- **prof** (*array like*) – Projected profile.

**Returns** Average profile in bins between the edges provided.

**Return type** numpy.array

### cluster_toolkit.bias module

Halo bias.

cluster_toolkit.bias.**bias_at_M**(*M*, *k*, *P*, *Omega_m*, *delta=200*)
Tinker et al. 2010 bais at mass M [Msun/h].

**Parameters**

- **M** (*float or array like*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.
- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.
- **Omega_m** (*float*) – Matter density fraction.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Halo bias.

**Return type** float or array like

cluster_toolkit.bias.**bias_at_R**(*R*, *k*, *P*, *delta=200*)
Tinker 2010 bais at mass M [Msun/h] corresponding to radius R [Mpc/h comoving].

**Parameters**

- **R** (*float or array like*) – Lagrangian radius in Mpc/h comoving.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.
- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Halo bias.

**Return type** float or array like

cluster_toolkit.bias.**bias_at_nu**(*nu*, *delta=200*)
Tinker 2010 bais at peak height nu.

**Parameters**

- **nu** (*float or array like*) – Peak height.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Halo bias.

**Return type** float or array like

cluster_toolkit.bias.**dbiasdM_at_M**(*M*, *k*, *P*, *Omega_m*, *delta=200*)
d/dM of Tinker et al. 2010 bais at mass M [Msun/h].

**Parameters**

- **M** (*float or array like*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.

- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.
- **Omega_m** (*float*) – Matter density fraction.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Derivative of the halo bias.

**Return type** float or array like

### cluster_toolkit.boostfactors module

Galaxy cluster boost factors, also known as membership dilution models.

cluster_toolkit.boostfactors.**boost_nfw_at_R**(*R*, *B0*, *R_scale*)
NFW boost factor model.

**Parameters**

- **R** (*float or array like*) – Distances on the sky in the same units as R_scale. Mpc/h comoving suggested for consistency with other modules.
- **B0** (*float*) – NFW profile amplitude.
- **R_scale** (*float*) – NFW profile scale radius.

**Returns** NFW boost factor profile; B = (1-fcl)^-1.

**Return type** float or array like

cluster_toolkit.boostfactors.**boost_powerlaw_at_R**(*R*, *B0*, *R_scale*, *alpha*)
Power law boost factor model.

**Parameters**

- **R** (*float or array like*) – Distances on the sky in the same units as R_scale. Mpc/h comoving suggested for consistency with other modules.
- **B0** (*float*) – Boost factor amplitude.
- **R_scale** (*float*) – Power law scale radius.
- **alpha** (*float*) – Power law exponent.

**Returns** Power law boost factor profile; B = (1-fcl)^-1.

**Return type** float or array like

### cluster_toolkit.concentration module

Halo concentration.

cluster_toolkit.concentration.**concentration_at_M**(*Mass*, *k*, *P*, *n_s*, *Omega_b*, *Omega_m*, *h*, *T_CMB=2.7255*, *delta=200*, *Mass_type='crit'*)
Concentration of the NFW profile at mass M [Msun/h]. Only implemented relation at the moment is Diemer & Kravtsov (2015).

Note: only single concentrations at a time are allowed at the moment.

**Parameters**

- **Mass** (*float*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.

- **P** (*array like*) – Linear matter power spectrum in (Mpc/h)^3 comoving.
- **n_s** (*float*) – Power spectrum tilt.
- **Omega_b** (*float*) – Baryonic matter density fraction.
- **Omega_m** (*float*) – Matter density fraction.
- **h** (*float*) – Reduced Hubble constant.
- **T_CMB** (*float*) – CMB temperature in Kelvin, default is 2.7.
- **delta** (*int; optional*) – Overdensity, default is 200.
- **Mass_type** (*string; optional*) –

**Returns** NFW concentration.

**Return type** float

### cluster_toolkit.deltasigma module

Galaxy cluster shear and magnification profiles also known as DeltaSigma and Sigma, respectively.

cluster_toolkit.deltasigma.**DeltaSigma_at_R**(*R*, *Rs*, *Sigma*, *mass*, *concentration*, *Omega_m*, *delta=200*)

Excess surface mass density given Sigma [Msun h/pc^2 comoving].

**Parameters**

- **R** (*float or array like*) – Projected radii Mpc/h comoving.
- **Rs** (*array like*) – Projected radii of Sigma, the surface mass density.
- **Sigma** (*array like*) – Surface mass density.
- **mass** (*float*) – Halo mass Msun/h.
- **concentration** (*float*) – concentration.
- **Omega_m** (*float*) – Matter density fraction.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Excess surface mass density Msun h/pc^2 comoving.

**Return type** float or array like

cluster_toolkit.deltasigma.**Sigma_at_R**(*R*, *Rxi*, *xi*, *mass*, *concentration*, *Omega_m*, *delta=200*)

Surface mass density given some 3d profile [Msun h/pc^2 comoving].

**Parameters**

- **R** (*float or array like*) – Projected radii Mpc/h comoving.
- **Rxi** (*array like*) – 3D radii of xi_hm Mpc/h comoving.
- **xi_hm** (*array like*) – Halo matter correlation function.
- **mass** (*float*) – Halo mass Msun/h.
- **concentration** (*float*) – concentration.
- **Omega_m** (*float*) – Matter density fraction.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Surface mass density Msun h/pc^2 comoving.

**Return type** float or array like

cluster_toolkit.deltasigma.**Sigma_nfw_at_R**(*R*, *mass*, *concentration*, *Omega_m*, *delta=200*)
Surface mass density of an NFW profile [Msun h/pc^2 comoving].

**Parameters**

- **R** (*float or array like*) – Projected radii Mpc/h comoving.
- **mass** (*float*) – Halo mass Msun/h.
- **concentration** (*float*) – concentration.
- **Omega_m** (*float*) – Matter density fraction.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Surface mass density Msun h/pc^2 comoving.

**Return type** float or array like

### cluster_toolkit.density module

Galaxy cluster density profiles.

cluster_toolkit.density.**rho_einasto_at_r**(*r*, *M*, *rs*, *alpha*, *Omega_m*, *delta=200*, *rhos=-1.0*)
Einasto halo density profile. Distances are Mpc/h comoving.

**Parameters**

- **r** (*float or array like*) – 3d distances from halo center.
- **M** (*float*) – Mass in Msun/h; not used if rhos is specified.
- **rhos** (*float*) – Scale density in Msun h^2/Mpc^3 comoving; optional.
- **rs** (*float*) – Scale radius.
- **alpha** (*float*) – Profile exponent.
- **Omega_m** (*float*) – Omega_matter, matter fraction of the density.
- **delta** (*int*) – Overdensity, default is 200.

**Returns** Einasto halo density profile in Msun h^2/Mpc^3 comoving.

**Return type** float or array like

cluster_toolkit.density.**rho_nfw_at_r**(*r*, *M*, *c*, *Omega_m*, *delta=200*)
NFW halo density profile.

**Parameters**

- **r** (*float or array like*) – 3d distances from halo center in Mpc/h comoving.
- **M** (*float*) – Mass in Msun/h.
- **c** (*float*) – Concentration.
- **Omega_m** (*float*) – Omega_matter, matter fraction of the density.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** NFW halo density profile in Msun h^2/Mpc^3 comoving.

**Return type** float or array like

### cluster_toolkit.exclusion module

Correlation functions with halo exclusion.

cluster_toolkit.exclusion.**theta_at_r**(*radii*, *rt*, *beta*)
Truncation function.

**Parameters**

- **radii** (*float or array-like*) – Radii of the profile in Mpc/h
- **rt** (*float*) – truncation radius in Mpc/h
- **beta** (*float*) – width of the truncation distribution (erfc) in Mpc/h

**Returns** Truncation function

**Return type** float or array-like

cluster_toolkit.exclusion.**xi_1h_exclusion_at_r**(*radii*, *Mass*, *conc*, *alpha*, *rt*, *beta*, *Omega_m*, *delta=200*)
Halo-matter correlation function with halo exclusion incorporated, but just the 1-halo term.

**Parameters**

- **radii** (*float or array-like*) – Radii of the profile in Mpc/h
- **Mass** (*float*) – Mass in Msun/h
- **conc** (*float*) – concentration of the 1-halo profile
- **alpha** (*float*) – Einasto parameter
- **rt** (*float*) – truncation radius in Mpc/h
- **beta** (*float*) – width of the truncation distribution (erfc) in Mpc/h
- **Omega_m** (*float*) – Matter density fraction
- **delta** (*float*) – halo overdensity; default is 200

**Returns** 1-halo of the exclusion profile at each radii

**Return type** float or array-like

cluster_toolkit.exclusion.**xi_2h_exclusion_at_r**(*radii*, *r_eff*, *beta_eff*, *bias*, *xi_mm*)
2-halo term in the halo-matter correlation function using halo exclusion theory.

**Parameters**

- **radii** (*float or array-like*) – Radii of the profile in Mpc/h
- **r_eff** (*float*) – effective radius for 2-halo subtraction in Mpc/h
- **beta_eff** (*float*) – width for effective radius truncation
- **bias** (*float*) – halo bias at large scales
- **xi_mm** (*float or array-like*) – matter correlation function. Must have same shape as the radii.

**Returns** 2-halo of the exclusion profile at each radii

**Return type** float or array-like

cluster_toolkit.exclusion.**xi_C_at_r**(*radii*, *r_A*, *r_B*, *beta_ex*, *xi_2h*)
Halo-matter correlation function with halo exclusion incorporated.

**Parameters**

- **radii** (*float or array-like*) – Radii of the profile in Mpc/h
- **r_A** (*float*) – radius of first correction term in Mpc/h
- **r_B** (*float*) – radius of second correction term in Mpc/h
- **beta_ex** (*float*) – width parameter for exclusion terms
- **xi_2h** (*float or array-like*) – 2-halo term of the exclusion profile

**Returns** correction term for the exclusion profile

**Return type** float or array-like

cluster_toolkit.exclusion.**xi_hm_exclusion_at_r**(*radii*, *Mass*, *conc*, *alpha*, *rt*, *beta*, *r_eff*, *beta_eff*, *r_A*, *r_B*, *beta_ex*, *bias*, *xi_mm*, *Omega_m*, *delta=200*)

Halo-matter correlation function with halo exclusion incorporated.

**Parameters**

- **radii** (*float or array-like*) – Radii of the profile in Mpc/h
- **Mass** (*float*) – Mass in Msun/h
- **conc** (*float*) – concentration of the 1-halo profile
- **alpha** (*float*) – Einasto parameter
- **rt** (*float*) – truncation radius in Mpc/h
- **beta** (*float*) – width of the truncation distribution (erfc) in Mpc/h
- **r_eff** (*float*) – effective radius for 2-halo subtraction in Mpc/h
- **beta_eff** (*float*) – width for effective radius truncation
- **r_A** (*float*) – radius of first correction term in Mpc/h
- **r_B** (*float*) – radius of second correction term in Mpc/h
- **beta_ex** (*float*) – width parameter for exclusion terms
- **bias** (*float*) – linear halo bias
- **xi_mm** (*float or array-like*) – matter correlation function. same shape as radii
- **Omega_m** (*float*) – matter density fraction
- **delta** (*int*) – halo overdensity. Default is 200

**Returns** exclusion profile at each radii

**Return type** float or array-like

### cluster_toolkit.massfunction module

Halo mass function.

cluster_toolkit.massfunction.**G_at_M**(*M*, *k*, *P*, *Omega_m*, *d=1.97*, *e=1.0*, *f=0.51*, *g=1.228*)

Tinker et al. 2008 appendix C multiplicity funciton G(M) as a function of mass. Default behavior is for $M_{200m}$ mass definition.

**Parameters**

- **M** (*float or array like*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of the matter power spectrum in h/Mpc comoving.

- **P_lin** (*array like*) – Linear matter power spectrum in (Mpc/h)^3 comoving.
- **Omega_m** (*float*) – Matter density fraction.
- **d** (*float; optional*) – First Tinker parameter. Default is 1.97.
- **e** (*float; optional*) – Second Tinker parameter. Default is 1.
- **f** (*float; optional*) – Third Tinker parameter. Default is 0.51.
- **g** (*float; optional*) – Fourth Tinker parameter. Default is 1.228.

**Returns** Halo multiplicity $G(M)$.

**Return type** float or array like

cluster_toolkit.massfunction.**G_at_sigma**(*sigma*, *d=1.97*, *e=1.0*, *f=0.51*, *g=1.228*)

Tinker et al. 2008 appendix C multiplicity funciton G(sigma) as a function of sigma.

NOTE: by default, this function is only valid at $z = 0$. For use at higher redshifts either recompute the parameters yourself, or wait for this behavior to be patched.

**Parameters**

- **sigma** (*float or array like*) – RMS variance of the matter density field.
- **d** (*float; optional*) – First Tinker parameter. Default is 1.97.
- **e** (*float; optional*) – Second Tinker parameter. Default is 1.
- **f** (*float; optional*) – Third Tinker parameter. Default is 0.51.
- **g** (*float; optional*) – Fourth Tinker parameter. Default is 1.228.

**Returns** Halo multiplicity G(sigma).

**Return type** float or array like

cluster_toolkit.massfunction.**dndM_at_M**(*M*, *k*, *P*, *Omega_m*, *d=1.97*, *e=1.0*, *f=0.51*, *g=1.228*)

Tinker et al. 2008 appendix C mass function at a given mass. Default behavior is for $M_{200m}$ mass definition.

NOTE: by default, this function is only valid at $z = 0$. For use at higher redshifts either recompute the parameters yourself, or wait for this behavior to be patched.

**Parameters**

- **M** (*float or array like*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of the matter power spectrum in h/Mpc comoving.
- **P_lin** (*array like*) – Linear matter power spectrum in (Mpc/h)^3 comoving.
- **Omega_m** (*float*) – Matter density fraction.
- **d** (*float; optional*) – First Tinker parameter. Default is 1.97.
- **e** (*float; optional*) – Second Tinker parameter. Default is 1.
- **f** (*float; optional*) – Third Tinker parameter. Default is 0.51.
- **g** (*float; optional*) – Fourth Tinker parameter. Default is 1.228.

**Returns** Mass function $dn/dM$.

**Return type** float or array like

cluster_toolkit.massfunction.**n_in_bin**(*Mlo*, *Mhi*, *Marr*, *dndM*)

Tinker et al. 2008 appendix C binned mass function.

**Parameters**

- **Mlo** (*float*) – Lower mass edge.
- **Mhi** (*float*) – Upper mass edge.
- **Marr** (*array like*) – Array of locations that dndM has been evaluated at.
- **dndM** (*array like*) – Array of dndM.

**Returns** number density of halos in the mass bin.

**Return type** float

cluster_toolkit.massfunction.**n_in_bins**(*edges*, *Marr*, *dndM*)
Tinker et al. 2008 appendix C binned mass function.

**Parameters**

- **edges** (*array like*) – Edges of the mass bins.
- **Marr** (*array like*) – Array of locations that dndM has been evaluated at.
- **dndM** (*array like*) – Array of dndM.

**Returns** number density of halos in the mass bins. Length is `len(edges)-1`.

**Return type** numpy.ndarray

### cluster_toolkit.miscentering module

Miscentering effects for projected profiles.

cluster_toolkit.miscentering.**DeltaSigma_mis_at_R**(*R*, *Rsigma*, *Sigma_mis*)
Miscentered excess surface mass density profile at R. Units are Msun h/pc^2 comoving.

**Parameters**

- **R** (*float or array like*) – Projected radii to evaluate profile.
- **Rsigma** (*array like*) – Projected radii of miscentered Sigma profile.
- **Sigma_mis** (*array like*) – Miscentered Sigma profile.

**Returns** Miscentered excess surface mass density profile.

**Return type** float array like

cluster_toolkit.miscentering.**Sigma_mis_at_R**(*R*, *Rsigma*, *Sigma*, *M*, *conc*, *Omega_m*, *Rmis*, *delta=200*, *kernel='rayleigh'*)
Miscentered surface mass density [Msun h/pc^2 comoving] convolved with a distribution for Rmis. Units are Msun h/pc^2 comoving.

**Parameters**

- **R** (*float or array like*) – Projected radii Mpc/h comoving.
- **Rsigma** (*array like*) – Projected radii of the centered surface mass density profile.
- **Sigma** (*float or array like*) – Surface mass density Msun h/pc^2 comoving.
- **M** (*float*) – Halo mass Msun/h.
- **conc** (*float*) – concentration.
- **Omega_m** (*float*) – Matter density fraction.
- **Rmis** (*float*) – Miscentered distance in Mpc/h comoving.

- **delta** (*int; optional*) – Overdensity, default is 200.
- **kernel** (*string; optional*) – Kernal for convolution. Options: rayleigh or gamma.

**Returns** Miscentered projected surface mass density.

**Return type** float or array like

cluster_toolkit.miscentering.**Sigma_mis_single_at_R**(*R*, *Rsigma*, *Sigma*, *M*, *conc*, *Omega_m*, *Rmis*, *delta=200*)

Miscentered surface mass density [Msun h/pc^2 comoving] of a profile miscentered by an amount Rmis Mpc/h comoving. Units are Msun h/pc^2 comoving.

**Parameters**

- **R** (*float or array like*) – Projected radii Mpc/h comoving.
- **Rsigma** (*array like*) – Projected radii of the centered surface mass density profile.
- **Sigma** (*float or array like*) – Surface mass density Msun h/pc^2 comoving.
- **M** (*float*) – Halo mass Msun/h.
- **conc** (*float*) – concentration.
- **Omega_m** (*float*) – Matter density fraction.
- **Rmis** (*float*) – Miscentered distance in Mpc/h comoving.
- **delta** (*int; optional*) – Overdensity, default is 200.

**Returns** Miscentered projected surface mass density.

**Return type** float or array like

### cluster_toolkit.peak_height module

Integrals of the power spectrum. This includes RMS variance of the density field, sigma2, as well as peak neight, nu. These were previously implemented in the bias module, but have been migrated to here.

cluster_toolkit.peak_height.**dsigma2dM_at_M**(*M*, *k*, *P*, *Omega_m*)

Derivative w.r.t. mass of RMS variance in top hat sphere of lagrangian radius R [Mpc/h comoving] corresponding to a mass M [Msun/h] of linear power spectrum.

**Parameters**

- **M** (*float or array like*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.
- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.
- **Omega_m** (*float*) – Omega_matter, matter density fraction.

**Returns** d/dM of RMS variance of top hat sphere.

**Return type** float or array like

cluster_toolkit.peak_height.**nu_at_M**(*M*, *k*, *P*, *Omega_m*)

Peak height of top hat sphere of lagrangian radius R [Mpc/h comoving] corresponding to a mass M [Msun/h] of linear power spectrum.

**Parameters**

- **M** (*float or array like*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.

- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.
- **Omega_m** (*float*) – Omega_matter, matter density fraction.

**Returns** Peak height.

**Return type** nu (float or array like)

cluster_toolkit.peak_height.**nu_at_R**(*R*, *k*, *P*)
Peak height of top hat sphere of radius R [Mpc/h comoving] of linear power spectrum.

**Parameters**

- **R** (*float or array like*) – Radius in Mpc/h comoving.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.
- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.

**Returns** Peak height.

**Return type** float or array like

cluster_toolkit.peak_height.**sigma2_at_M**(*M*, *k*, *P*, *Omega_m*)
RMS variance in top hat sphere of lagrangian radius R [Mpc/h comoving] corresponding to a mass M [Msun/h] of linear power spectrum.

**Parameters**

- **M** (*float or array like*) – Mass in Msun/h.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.
- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.
- **Omega_m** (*float*) – Omega_matter, matter density fraction.

**Returns** RMS variance of top hat sphere.

**Return type** float or array like

cluster_toolkit.peak_height.**sigma2_at_R**(*R*, *k*, *P*)
RMS variance in top hat sphere of radius R [Mpc/h comoving] of linear power spectrum.

**Parameters**

- **R** (*float or array like*) – Radius in Mpc/h comoving.
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving.
- **P** (*array like*) – Power spectrum in (Mpc/h)^3 comoving.

**Returns** RMS variance of a top hat sphere.

**Return type** float or array like

### cluster_toolkit.profile_derivatives module

Derivatives of halo profiles. Used to plot splashback results.

cluster_toolkit.profile_derivatives.**drho_nfw_dr_at_R**(*Radii*, *Mass*, *conc*, *Omega_m*, *delta=200*)
Derivative of the NFW halo density profile.

**Parameters**

- **Radii** (*float or array like*) – 3d distances from halo center in Mpc/h comoving

- **Mass** (*float*) – Mass in Msun/h
- **conc** (*float*) – Concentration
- **Omega_m** (*float*) – Matter fraction of the density
- **delta** (*int; optional*) – Overdensity, default is 200

**Returns** derivative of the NFW profile.

**Return type** float or array like

### cluster_toolkit.xi module

Correlation functions for matter and halos.

cluster_toolkit.xi.**xi_2halo**(*bias*, *xi_mm*)
2-halo term in halo-matter correlation function

**Parameters**

- **bias** (*float*) – Halo bias
- **xi_mm** (*float or array like*) – Matter-matter correlation function

**Returns** 2-halo term in halo-matter correlation function

**Return type** float or array like

cluster_toolkit.xi.**xi_DK**(*r*, *M*, *conc*, *be*, *se*, *k*, *P*, *om*, *delta=200*, *rhos=-1.0*, *alpha=-1.0*, *beta=-1.0*, *gamma=-1.0*)
Diemer-Kravtsov 2014 profile.

**Parameters**

- **r** (*float or array like*) – radii in Mpc/h comoving
- **M** (*float*) – mass in Msun/h
- **conc** (*float*) – Einasto concentration
- **be** (*float*) – DK transition parameter
- **se** (*float*) – DK transition parameter
- **k** (*array like*) – wavenumbers in h/Mpc
- **P** (*array like*) – matter power spectrum in [Mpc/h]^3
- **Omega_m** (*float*) – matter density fraction
- **delta** (*float*) – overdensity of matter. Optional, default is 200
- **rhos** (*float*) – Einasto density. Optional, default is compute from the mass
- **alpha** (*float*) – Einasto parameter. Optional, default is computed from peak height
- **beta** (*float*) – DK 2-halo parameter. Optional, default is 4
- **gamma** (*float*) – DK 2-halo parameter. Optional, default is 8

**Returns** DK profile evaluated at the input radii

**Return type** float or array like

cluster_toolkit.xi.**xi_DK_appendix1**(*r*, *M*, *conc*, *be*, *se*, *k*, *P*, *om*, *bias*, *xi_mm*, *delta=200*, *rhos=-1.0*, *alpha=-1.0*, *beta=-1.0*, *gamma=-1.0*)
Diemer-Kravtsov 2014 profile, first form from the appendix, eq. A3.

**Parameters**

- **r** (*float or array like*) – radii in Mpc/h comoving
- **M** (*float*) – mass in Msun/h
- **conc** (*float*) – Einasto concentration
- **be** (*float*) – DK transition parameter
- **se** (*float*) – DK transition parameter
- **k** (*array like*) – wavenumbers in h/Mpc
- **P** (*array like*) – matter power spectrum in [Mpc/h]^3
- **Omega_m** (*float*) – matter density fraction
- **bias** (*float*) – halo bias
- **xi_mm** (*float or array like*) – matter correlation function at r
- **delta** (*float*) – overdensity of matter. Optional, default is 200
- **rhos** (*float*) – Einasto density. Optional, default is compute from the mass
- **alpha** (*float*) – Einasto parameter. Optional, default is computed from peak height
- **beta** (*float*) – DK 2-halo parameter. Optional, default is 4
- **gamma** (*float*) – DK 2-halo parameter. Optional, default is 8

**Returns** DK profile evaluated at the input radii

**Return type** float or array like

cluster_toolkit.xi.**xi_DK_appendix2**(*r*, *M*, *conc*, *be*, *se*, *k*, *P*, *om*, *bias*, *xi_mm*, *delta=200*, *rhos=-1.0*, *alpha=-1.0*, *beta=-1.0*, *gamma=-1.0*)

Diemer-Kravtsov 2014 profile, second form from the appendix, eq. A4.

**Parameters**

- **r** (*float or array like*) – radii in Mpc/h comoving
- **M** (*float*) – mass in Msun/h
- **conc** (*float*) – Einasto concentration
- **be** (*float*) – DK transition parameter
- **se** (*float*) – DK transition parameter
- **k** (*array like*) – wavenumbers in h/Mpc
- **P** (*array like*) – matter power spectrum in [Mpc/h]^3
- **Omega_m** (*float*) – matter density fraction
- **bias** (*float*) – halo bias
- **xi_mm** (*float or array like*) – matter correlation function at r
- **delta** (*float*) – overdensity of matter. Optional, default is 200
- **rhos** (*float*) – Einasto density. Optional, default is compute from the mass
- **alpha** (*float*) – Einasto parameter. Optional, default is computed from peak height
- **beta** (*float*) – DK 2-halo parameter. Optional, default is 4
- **gamma** (*float*) – DK 2-halo parameter. Optional, default is 8

**Returns** DK profile evaluated at the input radii

**Return type** float or array like

cluster_toolkit.xi.**xi_einasto_at_r**(*r*, *M*, *conc*, *alpha*, *om*, *delta=200*, *rhos=-1.0*)
Einasto halo profile.

**Parameters**

- **r** (*float or array like*) – 3d distances from halo center in Mpc/h comoving
- **M** (*float*) – Mass in Msun/h; not used if rhos is specified
- **conc** (*float*) – Concentration
- **alpha** (*float*) – Profile exponent
- **om** (*float*) – Omega_matter, matter fraction of the density
- **delta** (*int*) – Overdensity, default is 200
- **rhos** (*float*) – Scale density in Msun h^2/Mpc^3 comoving; optional

**Returns** Einasto halo profile.

**Return type** float or array like

cluster_toolkit.xi.**xi_hm**(*xi_1halo*, *xi_2halo*, *combination='max'*)
Halo-matter correlation function

Note: at the moment you can combine the 1-halo and 2-halo terms by either taking the max of the two or the sum of the two. The 'combination' field must be set to either 'max' (default) or 'sum'.

**Parameters**

- **xi_1halo** (*float or array like*) – 1-halo term
- **xi_2halo** (*float or array like, same size as xi_1halo*) – 2-halo term
- **combination** (*string; optional*) – specifies how the 1-halo and 2-halo terms are combined, default is 'max' which takes the max of the two

**Returns** Halo-matter correlation function

**Return type** float or array like

cluster_toolkit.xi.**xi_mm_at_r**(*r*, *k*, *P*, *N=500*, *step=0.005*, *exact=False*)
Matter-matter correlation function.

**Parameters**

- **r** (*float or array like*) – 3d distances from halo center in Mpc/h comoving
- **k** (*array like*) – Wavenumbers of power spectrum in h/Mpc comoving
- **P** (*array like*) – Matter power spectrum in (Mpc/h)^3 comoving
- **N** (*int; optional*) – Quadrature step count, default is 500
- **step** (*float; optional*) – Quadrature step size, default is 5e-3
- **exact** (*boolean*) – Use the slow, exact calculation; default is False

**Returns** Matter-matter correlation function

**Return type** float or array like

cluster_toolkit.xi.**xi_nfw_at_r**(*r*, *M*, *c*, *Omega_m*, *delta=200*)
NFW halo profile correlation function.

**Parameters**

- **r** (*float or array like*) – 3d distances from halo center in Mpc/h comoving
- **M** (*float*) – Mass in Msun/h
- **c** (*float*) – Concentration
- **Omega_m** (*float*) – Omega_matter, matter fraction of the density
- **delta** (*int; optional*) – Overdensity, default is 200

**Returns** NFW halo profile.

**Return type** float or array like

### 11.1.2 Module contents

cluster_toolkit is a module for computing galaxy cluster models.

# CHAPTER 12

# Using CLASS

CLASS is a code used to compute the matter power spectrum. The power spectrum is a key input for the cluster-toolkit. This is meant to be a very short example of how you can call CLASS to get the linear and nonlinear power spectra.

**Note:** The CLASS github page is here. The CLASS documentation is found here.

**Note:** CLASS uses units of $Mpc^{-1}$ for math:$k$ and $Mpc^3$ for math:$P$.

```
from classy import Class
import numpy as np

#Start by specifying the cosmology
Omega_b = 0.05
Omega_m = 0.3
Omega_cdm = Omega_m - Omega_b
h = 0.7 #H0/100
A_s = 2.1e-9
n_s = 0.96

#Create a params dictionary
#Need to specify the max wavenumber
k_max = 10 #UNITS: 1/Mpc

params = {
              'output':'mPk',
              'non linear':'halofit',
              'Omega_b':Omega_b,
              'Omega_cdm':Omega_cdm,
              'h':h,
              'A_s':A_s,
```

(continues on next page)

(continued from previous page)

```
                'n_s':n_s,
                'P_k_max_1/Mpc':k_max,
                'z_max_pk':10. #Default value is 10
}

#Initialize the cosmology andcompute everything
cosmo = Class()
cosmo.set(params)
cosmo.compute()

#Specify k and z
k = np.logspace(-5, np.log10(k_max), num=1000) #Mpc^-1
z = 1.

#Call these for the nonlinear and linear matter power spectra
Pnonlin = np.array([cosmo.pk(ki, z) for ki in k])
Plin = np.array([cosmo.pk_lin(ki, z) for ki in k])

#NOTE: You will need to convert these to h/Mpc and (Mpc/h)^3
#to use in the toolkit. To do this you would do:
k /= h
Plin *= h**3
Pnonlin *= h**3
```

# CHAPTER 13

## Using CAMB

CAMB is similar to CLASS in that it is a Boltzmann code used to compute the matter power spectrum. Very recently, a Python wrapper was created for CAMB, however it is less documented and less modular compared to CLASS. However, for the sake of comparison you can follow this script to use CAMB to calculate the linear and nonlinear matter power spectra.

**Note:** CAMB and CLASS have differences that can cause >1% level changes in things like the mass function and possibly lensing. In general, pick one and make it explicit that you are using it when you describe your work.

**Note:** CAMB outputs tend to have different shapes than you would expect.

```
import camb
from camb import model, initialpower

#Set cosmological parameters
pars = camb.CAMBparams()
pars.set_cosmology(H0=67.5, ombh2=0.022, omch2=0.122)
pars.set_dark_energy(w=-1.0)
pars.InitPower.set_params(ns=0.965)

#This sets the k limits and specifies redshifts
pars.set_matter_power(redshifts=[0., 0.8], kmax=2.0)

#Linear P(k)
pars.NonLinear = model.NonLinear_none
results = camb.get_results(pars)
kh, z, pk = results.get_matter_power_spectrum(minkh=1e-4, maxkh=1, npoints = 1000)

#Note: the above function has the maxkh argument for specifying a different
#kmax than was used above.
#Note: pk has the shape (N_z, N_k)
```

(continues on next page)

(continued from previous page)

```
#Non-Linear spectra (Halofit)
pars.NonLinear = model.NonLinear_both
results.calc_power_spectra(pars)
khnl, znl, pknl = results.get_matter_power_spectrum(minkh=1e-4, maxkh=1, npoints =␣
↪1000)
```

# CHAPTER 14

# Frequently Asked Questions

## 14.1 I'm getting an interpolation error

The backend of the toolkit is written in C. It is likely that you are passing in data that in Python is actually of floating point precision (32 bits), when everything in C is expected to be in double precision (64 bits). You must do a cast in Python to correct this.

## 14.2 I'm getting random GSL errors

Python does not order arrays in memory the same way as in C. This can cause strange behavior as seemingly random memory addresses get used, and sometimes even segmentation faults. It is likely that your array in Python is not "C-ordered". Use this numpy function to force your input arrays to have the correct ordering in memory.

These two points are outstanding issues on the github issues page. One day they will be taken care of automatically without causing the code to slow significantly.

# Python Module Index

## C

# Index

## T



## X