skypy Documentation

Release 0.1

SkyPy Team

Apr 17, 2020

CONTENTS

I Getting Started 3

II User Documentation 5

III Developer Documentation 39

IV Project details 41

V Index 43

VI Acknowledgements 47

This package contains methods for modelling the Universe, galaxies and the Milky Way. Also included are methods for generating observed data.

Part I

Getting Started

Part II

User Documentation

CHAPTER

ONE

PACKAGES

1.1 Galaxies (skypy.galaxy)

This module contains methods that model the intrinsic properties of galaxy populations.

1.1.1 Luminosities (skypy.galaxy.luminosity)

Models of galaxy luminosities.

Models

herbel_luminosities	Model of Herbel et al (2017)

skypy.galaxy.luminosity.herbel_luminosities

skypy.galaxy.luminosity.**herbel_luminosities**(*redshift*, *alpha*, *a_m*, *b_m*, *size=None*, *x_min=0.00305*, *x_max=1100.0*, *resolution=100*)

Model of Herbel et al (2017)

Luminosities following the Schechter luminosity function following the Herbel et al. [1] model.

Parameters

redshift
[(nz,) array-like] The redshift values at which to sample luminosities.

alpha
[float or int] The alpha parameter in the Schechter luminosity function.

a_m, b_m
[float or int] Factors parameterising the characteristic absolute magnitude M_* as a linear function of redshift according to Equation 3.3 in [1].

size: int, optional
Output shape of luminosity samples. If size is None and redshift is a scalar, a single sample is returned. If size is None and redshift is an array, an array of samples is returned with the same shape as redshift.

x_min, x_max
[float or int, optional] Lower and upper luminosity bounds in units of L*.

resolution
[int, optional] Resolution of the inverse transform sampling spline. Default is 100.

Returns

luminosity
[array_like] Drawn luminosities from the Schechter luminosity function.

Notes

The Schechter luminosity function is given as

$$\Phi(L, z) = \frac{\Phi_\star(z)}{L_\star(z)} \left(\frac{L}{L_\star(z)}\right)^\alpha \exp\left(-\frac{L}{L_\star(z)}\right) .$$

Here the luminosity is defined as

$$L = 10^{-0.4M} ,$$

with absolute magnitude M. Furthermore, Herbel et al. [1] introduced

$$\Phi_\star(z) = b_\phi \exp(a_\phi z) , M_\star(z) = a_M z + b_M .$$

Now we have to rescale the Schechter function by the comoving element and get

$$\phi(L, z) = \frac{d_H d_M^2}{E(z)} \Phi(L, z) .$$

References

[1]

Examples

```
>>> import skypy.galaxy.luminosity as lum
```

Sample 100 luminosity values at redshift z = 1.0 with a_m = -0.9408582, b_m = -20.40492365, alpha = -1.3.

```
>>> luminosities = lum.herbel_luminosities(1.0, -1.3, -0.9408582,
...                                        -20.40492365, size=100)
```

Sample a luminosity value for every redshift in an array z with a_m = -0.9408582, b_m = -20.40492365, alpha = -1.3.

```
>>> z = np.linspace(0,2, 100)
>>> luminosities = lum.herbel_luminosities(z, -1.3, -0.9408582,
...                                        -20.40492365)
```

1.1.2 Ellipticities (skypy.galaxy.ellipticity)

Galaxy ellipticity module.

This module provides facilities to sample the ellipticities of galaxies.

Models

beta_ellipticity	Galaxy ellipticities sampled from a reparameterized beta distribution.

skypy.galaxy.ellipticity.beta_ellipticity

skypy.galaxy.ellipticity.**beta_ellipticity**(**args*, ***kwds*) = **<skypy.galaxy.ellipticity.beta_ellipticity_gen object>**

Galaxy ellipticities sampled from a reparameterized beta distribution.

The ellipticities follow a beta distribution parameterized by $e_{\rm ratio}$ and $e_{\rm sum}$ as presented in [1] Section III.A.

Parameters

e_ratio
[array_like] Mean ellipticity of the distribution, must be between 0 and 1.

e_sum
[array_like] Parameter controlling the width of the distribution, must be positive.

Notes

The probability distribution function $p(e)$ for ellipticity e is given by a beta distribution:

$$p(e) \sim \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}$$

for $0 <= e <= 1$, $a = e_{\rm sum} e_{\rm ratio}$, $b = e_{\rm sum}(1 - e_{\rm ratio})$, $0 < e_{\rm ratio} < 1$ and $e_{\rm sum} > 0$, where Γ is the gamma function.

References

[1]

Examples

```
>>> from skypy.galaxy.ellipticity import beta_ellipticity
```

Sample 10 random variates from the Kacprzak model with $e_{\rm ratio} = 0.5$ and $e_{\rm sum} = 1.0$:

```
>>> ellipticity = beta_ellipticity.rvs(0.5, 1.0, size=10)
```

Fix distribution parameters for repeated use:

```
>>> ellipticity_distribution = beta_ellipticity(0.5, 1.0)
>>> ellipticity_distribution.mean()
0.5
>>> ellipticity = ellipticity_distribution.rvs(size=10)
```

1.1.3 Redshifts (skypy.galaxy.redshift)

Galaxy redshift module.

This module provides facilities to sample galaxy redshifts using a number of models.

Models

herbel_redshift	Redshift following the Schechter luminosity function marginalised over luminosities following the Herbel et al.
herbel_pdf	Calculates the redshift pdf of the Schechter luminosity function according to the model of Herbel et al.
smail	Redshifts following the Smail et al.

skypy.galaxy.redshift.herbel_redshift

skypy.galaxy.redshift.**herbel_redshift**(*alpha*, *a_phi*, *b_phi*, *a_m*, *b_m*, *cosmology*, *low=0.0*, *high=2.0*, *size=None*, *absolute_magnitude_max=-16.0*, *resolution=100*)

Redshift following the Schechter luminosity function marginalised over luminosities following the Herbel et al. [1] model.

Parameters

alpha
[float or scalar] The alpha parameter in the Schechter luminosity function

a_phi, b_phi
[float or scalar] Parametrisation factors of the normalisation factor Phi_* as a function of redshift according to Herbel et al. [1] equation (3.4).

a_m, b_m
[float or scalar] Parametrisation factors of the characteristic absolute magnitude M_* as a function of redshift according to Herbel et al. [1] equation (3.3).

cosmology
[instance] Instance of an Astropy Cosmology class.

low
[float or array_like of floats, optional] The lower boundary of teh output interval. All values generated will be greater than or equal to low. The default value is 0.0.

high
[float or array_like of floats, optional] Upper boundary of output interval. All values generated will be less than high. The default value is 2.0

size
[int or tuple of ints, optional] The number of redshifts to sample. If None one value is

sampled.

absolute_magnitude_max
[float or scalar, optional] Upper limit of the considered absolute magnitudes of the galaxies to wanna sample.

resolution
[int, optional] Characterises the resolution of the sampling. Default is 100.

Returns

redshift_sample
[ndarray or float] Drawn redshifts from the marginalised Schechter luminosity function.

Notes

The Schechter luminosity function is given as

$$\Phi(L,z) = \frac{\Phi_\star(z)}{L_\star(z)} \left(\frac{L}{L_\star(z)}\right)^\alpha \exp\left(-\frac{L}{L_\star(z)}\right) .$$

Here the luminosity is defined as

$$L = 10^{-0.4M} ,$$

with absolute magnitude M. Furthermore, Herbel et al. [1] introduced

$$\Phi_\star(z) = b_\phi \exp(a_\phi z) , M_\star(z) = a_M z + b_M .$$

Now we have to rescale the Schechter function by the comoving element and get

$$\phi(L,z) = \frac{d_H d_M^2}{E(z)} \Phi(L,z) .$$

References

[1]

Examples

```
>>> from skypy.galaxy.redshift import herbel_redshift
>>> from astropy.cosmology import FlatLambdaCDM
```

Sample 100 redshift values from the Schechter luminosity function with a_m = -0.9408582, b_m = -20.40492365, a_phi = -0.10268436, b_phi = 0.00370253, alpha = -1.3.

```
>>> cosmology = FlatLambdaCDM(H0=70, Om0=0.3, Tcmb0=2.725)
>>> redshift = herbel_redshift(size=1000, low=0.01, alpha=-1.3,
...                     a_phi=-0.10268436,a_m=-0.9408582, b_phi=0.00370253,
...                     b_m=-20.40492365, cosmology=cosmology,
...                     absolute_magnitude_max=-16.)
```

skypy.galaxy.redshift.herbel_pdf

skypy.galaxy.redshift.**herbel_pdf**(*redshift*, *alpha*, *a_phi*, *b_phi*, *a_m*, *b_m*, *cosmology*, *luminosity_min*)

Calculates the redshift pdf of the Schechter luminosity function according to the model of Herbel et al. [1] equation (3.6).

That is, changing the absolute magnitude M in equation (3.2) to luminosity L, integrate over all possible L and multiplying by the comoving element using a flat ΛCDM model to get the corresponding pdf.

Parameters

redshift
[array_like] Input redshifts.

alpha
[float or scalar] The alpha parameter in the Schechter luminosity function

a_phi, b_phi
[float or scalar] Parametrisation factors of the normalisation factor Phi_* as a function of redshift according to Herbel et al. [1] equation (3.4).

a_m, b_m
[float or scalar] Parametrisation factors of the characteristic absolute magnitude M_* as a function of redshift according to Herbel et al. [1] equation (3.3).

cosmology
[instance] Instance of an Astropy Cosmology class.

luminosity_min
[float or scalar] Cut-off luminosity value such that the Schechter luminosity function diverges for L -> 0

Returns

pdf
[ndarray or float]

Un-normalised probability density function as a function of redshift

according to Herbel et al. [1].

Notes

This module calculates the function

$$\mathrm{pdf}(z) = \Phi_\star(z) \cdot \frac{d_H d_M^2}{E(z)} \cdot \Gamma\left(\alpha + 1, \frac{L_{\min}}{L_\star(z)}\right) ,$$

with $\Phi_\star(z) = b_\phi \exp(a_\phi z)$ and the second term the comoving element.

References

[1]

Examples

```
>>> from skypy.galaxy.redshift import herbel_pdf
>>> import skypy.utils.astronomy as astro
>>> from astropy.cosmology import FlatLambdaCDM
>>> import numpy as np
```

Calculate the pdf for 100 redshift values between 0 and 2 with a_m = -0.9408582, b_m = -20.40492365, a_phi = -0.10268436, b_phi = 0.00370253, alpha = -1.3 for a flat cosmology.

```
>>> cosmology = FlatLambdaCDM(H0=70, Om0=0.3, Tcmb0=2.725)
>>> redshift = np.linspace(0, 2, 100)
>>> luminosity_min = astro.luminosity_from_absolute_magnitude(-16.0)
>>> redshift = herbel_pdf(redshift=redshift, alpha=-1.3,
...                       a_phi=-0.10268436,a_m=-0.9408582, b_phi=0.00370253,
...                       b_m=-20.40492365, cosmology=cosmology,
...                       luminosity_min=luminosity_min)
```

skypy.galaxy.redshift.smail

skypy.galaxy.redshift.**smail**(**args*, ***kwds*) **= <skypy.galaxy.redshift.smail_gen object>**

Redshifts following the Smail et al. (1994) model.

The redshift follows the Smail et al. [1] redshift distribution.

Parameters

z_median
[float or array_like of floats] Median redshift of the distribution, must be positive.

alpha
[float or array_like of floats] Power law exponent (z/z0)^alpha, must be positive.

beta
[float or array_like of floats] Log-power law exponent exp[-(z/z0)^beta], must be positive.

Notes

The probability distribution function $p(z)$ for redshift z is given by Amara & Refregier [2] as

$$p(z) \sim \left(\frac{z}{z_0}\right)^{\alpha} \exp\left[-\left(\frac{z}{z_0}\right)^{\beta}\right] .$$

This is the generalised gamma distribution `scipy.stats.gengamma`.

References

[1], [2]

Examples

```
>>> from skypy.galaxy.redshift import smail
```

Sample 10 random variates from the Smail model with `alpha = 1.5` and `beta = 2` and median redshift `z_median = 1.2`.

```
>>> redshift = smail.rvs(1.2, 1.5, 2.0, size=10)
```

Fix distribution parameters for repeated use.

```
>>> redshift_dist = smail(1.2, 1.5, 2.0)
>>> redshift_dist.median()
1.2
>>> redshift = redshift_dist.rvs(size=10)
```

1.1.4 Sizes (skypy.galaxy.size)

Galaxy size module.

This modules computes the angular size of galaxies from their physical size.

Utility functions

angular_size	Angular size of a galaxy.

skypy.galaxy.size.angular_size

skypy.galaxy.size.**angular_size**(*physical_size*, *redshift*, *cosmology*)

Angular size of a galaxy. This function transforms physical radius into angular distance, described in [1].

Parameters

physical_size
[astropy.Quantity] Physical radius of galaxies in units of length.

redshift
[float] Redshifts at which to evaluate the angular diameter distance.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing methods for the evolution history of omega_matter and omega_lambda with redshift.

Returns

angular_size
[astropy.Quantity] Angular distances in units of [rad] for a given radius.

References

[1]

Examples

```
>>> from astropy import units
>>> from astropy.cosmology import FlatLambdaCDM
>>> cosmology = FlatLambdaCDM(H0=67.04, Om0=0.3183, Ob0=0.047745)
>>> size = 10.0 * units.kpc
>>> angular_size(size, 1, cosmology)
<Quantity 5.85990062e-06 rad>
```

Models

early_type_lognormal	Lognormal distribution for early-type galaxies.
late_type_lognormal	Lognormal distribution for late-type galaxies.
linear_lognormal	Lognormal distribution with linear mean.

skypy.galaxy.size.early_type_lognormal

skypy.galaxy.size.**early_type_lognormal**(*magnitude*, *a*, *b*, *M0*, *sigma1*, *sigma2*, *size=None*)

Lognormal distribution for early-type galaxies. This function provides a lognormal distribution for the physical size of early-type galaxies, described by equations 12, 14 and 16 in [1].

Parameters

magnitude
[float or array_like.] Galaxy magnitude at which evaluate the lognormal distribution.

a, b
[float] Linear model parameters describing the mean size of galaxies, equation 3.14 in [1].

sigma: float
Standard deviation of the lognormal distribution for the physical radius of galaxies.

size
[int or tuple of ints, optional.] Output shape. If the given shape is, e.g., (m, n, k), then m * n * k samples are drawn. If size is None (default), a single value is returned if mean and sigma are both scalars. Otherwise, np.broadcast(mean, sigma).size samples are drawn.

Returns

physical_size
[ndarray or astropy.Quantity] Physical distance for a given galaxy with a given magnitude, in [kpc]. If size is None and magnitude is a scalar, a single sample is returned. If size is ns, different from None, and magnitude is scalar, shape is (ns,). If magnitude has shape (nm,) and size=None, shape is (nm,).

References

..[1] S. Shen, H.J. Mo, S.D.M. White, M.R. Blanton, G. Kauffmann, W. Voges,

J. Brinkmann, I. Csabai, Mon. Not. Roy. Astron. Soc. 343, 978 (2003).

Examples

```
>>> import numpy as np
>>> from skypy.galaxy import size
>>> np.random.seed(12345)
>>> magnitude = -20.0
>>> a, b, M0 = 0.6, -4.63, -20.52
>>> sigma1, sigma2 = 0.48, 0.25
>>> size.early_type_lognormal(magnitude, a, b, M0, sigma1, sigma2)
<Quantity 1.35830285 kpc>
```

skypy.galaxy.size.late_type_lognormal

skypy.galaxy.size.**late_type_lognormal**(*magnitude*, *alpha*, *beta*, *gamma*, *M0*, *sigma1*, *sigma2*, *size=None*)

Lognormal distribution for late-type galaxies. This function provides a lognormal distribution for the physical size of late-type galaxies, described by equations 12, 15 and 16 in [1].

Parameters

magnitude
[float or array_like.] Galaxy magnitude at which evaluate the lognormal distribution.

alpha, beta, gamma, M0: float
Model parameters describing the mean size of galaxies in [kpc]. Equation 15 in [1].

sigma1, sigma2: float
Parameters describing the standard deviation of the lognormal distribution for the physical radius of galaxies. Equation 16 in [1].

size
[int or tuple of ints, optional.] Output shape. If the given shape is, e.g., (m, n, k), then m * n * k samples are drawn. If size is None (default), a single value is returned if mean and sigma are both scalars. Otherwise, np.broadcast(mean, sigma).size samples are drawn.

Returns

physical_size
[numpy.ndarray or astropy.Quantity] Physical distance for a given galaxy with a given magnitude, in [kpc]. If size is None and magnitude is a scalar, a single sample is returned. If size is ns, different from None, and magnitude is scalar, shape is (ns,). If magnitude has shape (nm,) and size=None, shape is (nm,).

References

..[1] S. Shen, H.J. Mo, S.D.M. White, M.R. Blanton, G. Kauffmann, W. Voges,

J. Brinkmann, I. Csabai, Mon. Not. Roy. Astron. Soc. 343, 978 (2003).

Examples

```
>>> import numpy as np
>>> from skypy.galaxy import size
>>> np.random.seed(12345)
>>> magnitude = -16.0
>>> alpha, beta, gamma, M0 = 0.21, 0.53, -1.31, -20.52
>>> sigma1, sigma2 = 0.48, 0.25
>>> size.late_type_lognormal(magnitude, alpha, beta, gamma, M0,
↪sigma1, sigma2)
<Quantity 0.9850926 kpc>
```

skypy.galaxy.size.linear_lognormal

skypy.galaxy.size.**linear_lognormal**(*magnitude*, *a_mu*, *b_mu*, *sigma*, *size=None*)

Lognormal distribution with linear mean. This function provides a lognormal distribution for the physical size of galaxies with a linear mean, described by equation 3.14 in [1]. See also equation 14 in [2].

Parameters

magnitude
[float or array_like.] Galaxy absolute magnitude at which evaluate the lognormal distribution.

a_mu, b_mu
[float] Linear model parameters describing the mean size of galaxies, equation 3.14 in [1].

sigma: float
Standard deviation of the lognormal distribution for the physical radius of galaxies.

size
[int or tuple of ints, optional.] Output shape. If the given shape is, e.g., (m, n, k), then m * n * k samples are drawn. If size is None (default), a single value is returned if mean and sigma are both scalars. Otherwise, np.broadcast(mean, sigma).size samples are drawn.

Returns

physical_size
[numpy.ndarray or astropy.Quantity] Physical distance for a given galaxy with a given magnitude, in [kpc]. If size is None and magnitude is a scalar, a single sample is returned. If size is ns, different from None, and magnitude is scalar, shape is (ns,). If magnitude has shape (nm,) and size=None, shape is (nm,).

References

..[1] J. Herbel, T. Kacprzak, A. Amara, A. Refregier, C.Bruderer and

A. Nicola, JCAP 1708, 035 (2017).

..[2] S. Shen, H.J. Mo, S.D.M. White, M.R. Blanton, G. Kauffmann, W.Voges,

J. Brinkmann, I.Csabai, Mon. Not. Roy. Astron. Soc. 343, 978 (2003).

Examples

```
>>> import numpy as np
>>> from skypy.galaxy import size
>>> np.random.seed(12345)
>>> magnitude = -20.0
>>> a_mu, b_mu, sigma =-0.24, -4.63, 0.4
>>> size.linear_lognormal(magnitude, a_mu, b_mu, sigma)
<Quantity 1.36282044 kpc>
```

1.1.5 Spectra (skypy.galaxy.spectrum)

Galaxy spectrum module.

Models

dirichlet_coefficients	Dirichlet-distributed SED coefficients.

skypy.galaxy.spectrum.dirichlet_coefficients

skypy.galaxy.spectrum.**dirichlet_coefficients**(*redshift*, *alpha0*, *alpha1*, *z1=1.0*)

Dirichlet-distributed SED coefficients.

Spectral coefficients to calculate the rest-frame spectral energy
distribution of a galaxy following the Herbel et al. model in [1].

Parameters

redshift
[(nz,) array_like] The redshift values of the galaxies for which the coefficients want to be sampled.

alpha0, alpha1
[(nc,) array_like] Factors parameterising the Dirichlet distribution according to Equation (3.9) in [1].

z1
[float or scalar, optional] Reference redshift at which alpha = alpha1. The default value is 1.0.

Returns

coefficients
[(nz, nc) ndarray] The spectral coefficients of the galaxies. The shape is (n, nc) with nz the number of redshifts and nc the number of coefficients.

Notes

One example is the rest-frame spectral energy distribution of galaxies which can be written as a linear combination of the five kcorrect [2] template spectra f_i (see [1])

$$f(\lambda) = \sum_{i=1}^{5} c_i f_i(\lambda) \ ,$$

where the coefficients c_i were shown to follow a Dirichlet distribution of order 5. The five parameters describing the Dirichlet distribution are given by

$$\alpha_i(z) = (\alpha_{i,0})^{1-z/z_1} \cdot (\alpha_{i,1})^{z/z_1} \ .$$

Here, $\alpha_{i,0}$ describes the galaxy population at redshift $z = 0$ and $\alpha_{i,1}$ the population at $z = z_1 > 0$. These parameters depend on the galaxy type and we chose $z_1 = 1$.

Beside this example, this code works for a general number of templates.

References

[1], [2]

Examples

```
>>> from skypy.galaxy.spectrum import dirichlet_coefficients
>>> import numpy as np
```

Sample the coefficients according to [1] for n blue galaxies with redshifts between 0 and 1.

```
>>> n = 100000
>>> alpha0 = np.array([2.079, 3.524, 1.917, 1.992, 2.536])
>>> alpha1 = np.array([2.265, 3.862, 1.921, 1.685, 2.480])
>>> redshift = np.linspace(0,2, n)
>>> coefficients = dirichlet_coefficients(redshift, alpha0, alpha1)
```

1.2 Dark Matter Halos (skypy.halo)

This module contains methods that model the properties of dark matter halo populations.

1.2.1 Abundance Matching (skypy.halo.abundance_matching)

Abundance matching module.

This module provides methods to perform abundance matching between catalogs of galaxies and dark matter halos.

Models

vale_ostriker	Vale & Ostriker abundance matching.

skypy.halo.abundance_matching.vale_ostriker

skypy.halo.abundance_matching.**vale_ostriker**(*halos*, *galaxies*, *mass='mass'*, *luminosity='luminosity'*, *join_type='inner'*)

Vale & Ostriker abundance matching. Takes catalogs of halos and galaxies and performs abundance matching following the method outlined in Vale & Ostriker (2004) assuming monotonicity between halo mass and galaxy luminosity to return an abundance-matched table of halo-galaxy pairs.

Parameters

halos, galaxies
[Astropy Table] Tables of halos and galaxies to be matched.

mass, luminosity
[str] Halo mass and galaxy luminosity column names.

join_type
[str] Join type ('inner' | 'exact' | 'outer'), default is inner

Returns

matched_Table
[Astropy Table] Table of abundance-matched halos and galaxies.

References

[1]

1.2.2 Mass (skypy.halo.mass)

Halo mass module.

This module provides methods to sample the masses of dark matter halos.

Models

press_schechter(n, m_star[, size, x_min, ...])	Sampling from Press-Schechter mass function (1974).

1.3 Power Spectrum (skypy.power_spectrum)

This module contains methods that model the matter power spectrum.

1.3.1 Linear Power Spectrum

camb	Return the CAMB computation of the linear matter power spectrum, on a two dimensional grid of wavenumber and redshift
eisenstein_hu	Eisenstein & Hu fitting function for the linear matter power spectrum with (or without) baryon acoustic oscillations using formulation from Komatsu et al (2009).
transfer_no_wiggles	Eisenstein & Hu fitting formula for the transfer function without baryon acoustic oscillation wiggles.
transfer_with_wiggles	Eisenstein & Hu fitting formula for the transfer function with baryon acoustic oscillation wiggles.

skypy.power_spectrum.camb

skypy.power_spectrum.**camb**(*wavenumber*, *redshift*, *cosmology*, *A_s*, *n_s*)

Return the CAMB computation of the linear matter power spectrum, on a two dimensional grid of wavenumber and redshift

Parameters

wavenumber
[(nk,) array_like] Array of wavenumbers in units of [Mpc^-1] at which to evaluate the linear matter power spectrum.

redshift
[(nz,) array_like] Array of redshifts at which to evaluate the linear matter power spectrum.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing omega_matter, omega_baryon, Hubble parameter and CMB temperature in the present day

A_s
[float] Cosmology parameter, amplitude normalisation of curvature perturbation power spectrum

n_s
[float] Cosmology parameter, spectral index of scalar perturbation power spectrum

Returns

power_spectrum
[(nk, nz) array_like] Array of values for the linear matter power spectrum in [Mpc^3] evaluated at the input wavenumbers for the given primordial power spectrum parameters, cosmology. For nz redshifts and nk wavenumbers the returned array will have shape (nk, nz).

References

doi : 10.1086/309179 arXiv: astro-ph/9911177

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import default_cosmology
>>> cosmology = default_cosmology.get()
>>> redshift = np.array([0, 1])
>>> wavenumber = np.array([1.e-2, 1.e-1, 1e0])
>>> A_s = 2.e-9
>>> n_s = 0.965
>>> camb(wavenumber, redshift, cosmology, A_s, n_s)
array([[2.34758952e+04, 8.70837957e+03],
       [3.03660813e+03, 1.12836115e+03],
       [2.53124880e+01, 9.40802814e+00]])
```

skypy.power_spectrum.eisenstein_hu

skypy.power_spectrum.**eisenstein_hu**(*wavenumber*, *A_s*, *n_s*, *cosmology*, *kwmap=0.02*, *wiggle=True*)
Eisenstein & Hu fitting function for the linear matter power spectrum with (or without) baryon acoustic oscillations using formulation from Komatsu et al (2009).

Parameters

wavenumber
[array_like] Array of wavenumbers in units of [Mpc^-1] at which to evaluate the linear matter power spectrum.

A_s, n_s: float
Amplitude and spectral index of primordial scalar fluctuations.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing omega_matter, omega_baryon, Hubble parameter and CMB temperature in the present day.

kwmap
[float] WMAP normalization for the amplitude of primordial scalar fluctuations, as described in [3], in units of [Mpc^-1]. Default is 0.02.

wiggle
[bool] Boolean flag to set the use of baryion acoustic oscillations wiggles. Default is True, for which the power spectrum is computed with the wiggles.

Returns

power_spectrum
[array_like] Linear matter power spectrum in units of [Mpc^3], evaluated at the given

wavenumbers for the input primordial power spectrum parameters A_s and n_s, cosmology, and kwmap normalization.

References

[1], [2], [3]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import default_cosmology
>>> wavenumber = np.logspace(-3, 1, num=5, base=10.0)
>>> A_s, n_s = 2.1982e-09, 0.969453
>>> cosmology = default_cosmology.get()
>>> eisenstein_hu(wavenumber, A_s, n_s, cosmology, kwmap=0.02,
...               wiggle=True)
array([6.47460158e+03, 3.71610099e+04, 9.65702614e+03, 1.14604456e+02,
   3.91399918e-01])
```

skypy.power_spectrum.transfer_no_wiggles

skypy.power_spectrum.**transfer_no_wiggles**(*wavenumber*, *A_s*, *n_s*, *cosmology*)

Eisenstein & Hu fitting formula for the transfer function without baryon acoustic oscillation wiggles.

Parameters

wavenumber
: [array_like] Array of wavenumbers in units of [Mpc^-1] at which to evaluate the linear matter power spectrum.

A_s, n_s: float
: Amplitude and spectral index of primordial scalar fluctuations.

cosmology
: [astropy.cosmology.Cosmology] Cosmology object providing omega_matter, omega_baryon, Hubble parameter and CMB temperature in the present day.

Returns

transfer
: [array_like] Transfer function evaluated at the given wavenumbers for the input primordial power spectrum parameters A_s and n_s, cosmology and kwmap normalization.

References

[1], [2], [3]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import default_cosmology
>>> wavenumber = np.logspace(-3, 1, num=5, base=10.0)
>>> A_s, n_s = 2.1982e-09, 0.969453
>>> cosmology = default_cosmology.get()
>>> transfer_no_wiggles(wavenumber, A_s, n_s, cosmology)
array([9.91959695e-01, 7.84518347e-01, 1.32327555e-01, 4.60773671e-03,
   8.78447096e-05])
```

skypy.power_spectrum.transfer_with_wiggles

skypy.power_spectrum.**transfer_with_wiggles**(*wavenumber*, *A_s*, *n_s*, *cosmology*, *kwmap=0.02*)

Eisenstein & Hu fitting formula for the transfer function with baryon acoustic oscillation wiggles.

Parameters

wavenumber
[array_like] Array of wavenumbers in units of [Mpc^-1] at which to evaluate the linear matter power spectrum.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing omega_matter, omega_baryon, Hubble parameter and CMB temperature at the present day.

A_s, n_s: float
Amplitude and spectral index of primordial scalar fluctuations.

kwmap
[float] WMAP normalization for the amplitude of primordial scalar fluctuations, as described in [3], in units of [Mpc^-1]. Default is 0.02.

Returns

transfer
[array_like] Transfer function evaluated at the given array of wavenumbers for the input primordial power spectrum parameters A_s and n_s, cosmology and kwmap normalization.

References

[1], [2], [3]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import default_cosmology
>>> wavenumber = np.logspace(-3, 1, num=5, base=10.0)
>>> A_s, n_s = 2.1982e-09, 0.969453
>>> cosmology = default_cosmology.get()
>>> transfer_with_wiggles(wavenumber, A_s, n_s, cosmology, kwmap=0.02)
array([9.92144790e-01, 7.78548704e-01, 1.29998169e-01, 4.63863054e-03,
   8.87918075e-05])
```

1.3.2 Nonlinear Power Spectrum

HalofitParameters	
halofit	Computation of the non-linear halo power spectrum.
halofit_smith	Computation of the non-linear halo power spectrum.
halofit_takahashi	Computation of the non-linear halo power spectrum.
halofit_bird	Computation of the non-linear halo power spectrum.

skypy.power_spectrum.HalofitParameters

class skypy.power_spectrum.**HalofitParameters**(*a*, *b*, *c*, *gamma*, *alpha*, *beta*, *mu*, *nu*, *fa*, *fb*, *l*, *m*, *p*, *r*, *s*, *t*)

Create new instance of HalofitParameters(a, b, c, gamma, alpha, beta, mu, nu, fa, fb, l, m, p, r, s, t)

__init__(*self*, */*, **args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(self, /, *args, **kwargs)	Initialize self.
count(self, value, /)	Return number of occurrences of value.
index(self, value[, start, stop])	Return first index of value.

Attributes

a	Alias for field number 0
alpha	Alias for field number 4
b	Alias for field number 1
beta	Alias for field number 5
c	Alias for field number 2
fa	Alias for field number 8
fb	Alias for field number 9
gamma	Alias for field number 3
l	Alias for field number 10
m	Alias for field number 11
mu	Alias for field number 6

Continued on next page

Table 12 – continued from previous page

nu	Alias for field number 7
p	Alias for field number 12
r	Alias for field number 13
s	Alias for field number 14
t	Alias for field number 15

skypy.power_spectrum.halofit

skypy.power_spectrum.**halofit**(*wavenumber*, *redshift*, *linear_power_spectrum*, *cosmology*, *parameters*)

Computation of the non-linear halo power spectrum.

This function computes the non-linear halo power spectrum, as a function of redshift and wavenumbers, following [1], [2] and [3].

Parameters

k
[(nk,) array_like] Input wavenumbers in units of [Mpc^-1].

z
[(nz,) array_like] Input redshifts

P
[(nk, nz) array_like] Linear power spectrum for given wavenumbers and redshifts [Mpc^3].

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing method for the evolution of omega_matter with redshift.

parameters
[HalofitParameters] namedtuple containing the free parameters of the model.

Returns

pknl
[(nk, nz) array_like] Non-linear halo power spectrum in units of [Mpc^3].

References

[1], [2], [3]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import FlatLambdaCDM
>>> kvec = np.array([1.00000000e-04, 1.01000000e+01])
>>> zvalue = 0.0
>>> pvec = np.array([388.6725682632502, 0.21676249605280398])
>>> cosmo = FlatLambdaCDM(H0=67.04, Om0=0.21479, Ob0=0.04895)
>>> halofit(kvec, zvalue, pvec, cosmo, _takahashi_parameters)
array([388.67064424,   0.72797614])
```

skypy.power_spectrum.halofit_smith

skypy.power_spectrum.**halofit_smith**(*wavenumber, redshift, linear_power_spectrum, cosmology, *, parameters=HalofitParameters(a=[0.167, 0.794, 1.6762, 1.8369, 1.4861, -0.6206, 0.0], b=[0.3084, 0.9466, 0.9463, -0.94, 0.0], c=[0.3214, 0.6669, -0.2807, -0.0793], gamma=[0.2989, 0.8649, 0.1631], alpha=[-0.1452, 0.37, 1.3884, 0.0], beta=[0.0, 0.0, 0.3401, 0.9854, 0.8291, 0.0], mu=[0.1908, -3.5442], nu=[1.2857, 0.9589], fa=[-0.0732, -0.1423, 0.0725], fb=[-0.0307, -0.0585, 0.0743], l=0.0, m=0.0, p=0.0, r=0.0, s=0.0, t=0.0)*)

Computation of the non-linear halo power spectrum.

This function computes the non-linear halo power spectrum, as a function of redshift and wavenumbers, following [1], [2] and [3].

Parameters

k
: [(nk,) array_like] Input wavenumbers in units of [Mpc^-1].

z
: [(nz,) array_like] Input redshifts

P
: [(nk, nz) array_like] Linear power spectrum for given wavenumbers and redshifts [Mpc^3].

cosmology
: [astropy.cosmology.Cosmology] Cosmology object providing method for the evolution of omega_matter with redshift.

parameters
: [HalofitParameters] namedtuple containing the free parameters of the model.

Returns

pknl
: [(nk, nz) array_like] Non-linear halo power spectrum in units of [Mpc^3].

References

[1], [2], [3]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import FlatLambdaCDM
>>> kvec = np.array([1.00000000e-04, 1.01000000e+01])
>>> zvalue = 0.0
>>> pvec = np.array([388.6725682632502, 0.21676249605280398])
>>> cosmo = FlatLambdaCDM(H0=67.04, Om0=0.21479, Ob0=0.04895)
>>> halofit(kvec, zvalue, pvec, cosmo, _takahashi_parameters)
array([388.67064424,   0.72797614])
```

skypy.power_spectrum.halofit_takahashi

skypy.power_spectrum.**halofit_takahashi**(*wavenumber, redshift, linear_power_spectrum, cosmology, *, parameters=HalofitParameters(a=[0.225, 0.9903, 2.3706, 2.8553, 1.5222, -0.6038, 0.1749], b=[0.5716, 0.5864, -0.5642, -1.5474, 0.2279], c=[0.8161, 2.0404, 0.3698, 0.5869], gamma=[-0.0843, 0.1971, 0.846], alpha=[-0.1959, 1.3373, 6.0835, -5.5274], beta=[0.398, 1.249, 0.3157, -0.7354, 2.0379, -0.1682], mu=[0.0, -inf], nu=[3.6902, 5.2105], fa=[-0.0732, -0.1423, 0.0725], fb=[-0.0307, -0.0585, 0.0743], l=0.0, m=0.0, p=0.0, r=0.0, s=0.0, t=0.0)*)

Computation of the non-linear halo power spectrum.

This function computes the non-linear halo power spectrum, as a function of redshift and wavenumbers, following [1], [2] and [3].

Parameters

k
: [(nk,) array_like] Input wavenumbers in units of [Mpc^-1].

z
: [(nz,) array_like] Input redshifts

P
: [(nk, nz) array_like] Linear power spectrum for given wavenumbers and redshifts [Mpc^3].

cosmology
: [astropy.cosmology.Cosmology] Cosmology object providing method for the evolution of omega_matter with redshift.

parameters
: [HalofitParameters] namedtuple containing the free parameters of the model.

Returns

pknl
: [(nk, nz) array_like] Non-linear halo power spectrum in units of [Mpc^3].

References

[1], [2], [3]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import FlatLambdaCDM
>>> kvec = np.array([1.00000000e-04, 1.01000000e+01])
>>> zvalue = 0.0
>>> pvec = np.array([388.6725682632502, 0.21676249605280398])
>>> cosmo = FlatLambdaCDM(H0=67.04, Om0=0.21479, Ob0=0.04895)
>>> halofit(kvec, zvalue, pvec, cosmo, _takahashi_parameters)
array([388.67064424,   0.72797614])
```

skypy.power_spectrum.halofit_bird

skypy.power_spectrum.**halofit_bird**(*wavenumber, redshift, linear_power_spectrum, cosmology, *, parameters=HalofitParameters(a=[0.167, 0.794, 1.6762, 1.8369, 1.4861, -0.6206, 0.0], b=[0.3084, 0.9466, 0.9463, -0.94, 0.0], c=[0.3214, 0.6669, -0.2807, -0.0793], gamma=[0.2224, 1.18075, -0.6719], alpha=[-0.1452, 0.37, 1.3884, 0.0], beta=[0.0, 0.0, 0.3401, 0.9854, 0.8291, 0.0], mu=[0.1908, -3.5442], nu=[1.2857, 0.9589], fa=[-0.0732, -0.1423, 0.0725], fb=[-0.0307, -0.0585, 0.0743], l=2.08, m=0.0012, p=26.3, r=-6.49, s=1.44, t=12.4)*)

Computation of the non-linear halo power spectrum.

This function computes the non-linear halo power spectrum, as a function of redshift and wavenumbers, following [1], [2] and [3].

Parameters

k
: [(nk,) array_like] Input wavenumbers in units of [Mpc^-1].

z
: [(nz,) array_like] Input redshifts

P
: [(nk, nz) array_like] Linear power spectrum for given wavenumbers and redshifts [Mpc^3].

cosmology
: [astropy.cosmology.Cosmology] Cosmology object providing method for the evolution of omega_matter with redshift.

parameters
: [HalofitParameters] namedtuple containing the free parameters of the model.

Returns

pknl
: [(nk, nz) array_like] Non-linear halo power spectrum in units of [Mpc^3].

References

[1], [2], [3]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import FlatLambdaCDM
>>> kvec = np.array([1.00000000e-04, 1.01000000e+01])
>>> zvalue = 0.0
>>> pvec = np.array([388.6725682632502, 0.21676249605280398])
>>> cosmo = FlatLambdaCDM(H0=67.04, Om0=0.21479, Ob0=0.04895)
>>> halofit(kvec, zvalue, pvec, cosmo, _takahashi_parameters)
array([388.67064424,   0.72797614])
```

1.3.3 Growth Functions

growth_factor	Computation of the growth factor.
growth_function	Computation of the growth function.
growth_function_carroll	Computation of the growth function.
growth_function_derivative	Computation of the first derivative of the growth function.

skypy.power_spectrum.growth_factor

skypy.power_spectrum.**growth_factor**(*redshift*, *cosmology*, *gamma=0.5454545454545454*)

Computation of the growth factor.

Function used to calculate f(z), parametrised growth factor at different redshifts, as described in [1] equation 17.

Parameters

redshift
[array_like] Array of redshifts at which to evaluate the growth function.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing methods for the evolution history of omega_matter and omega_lambda with redshift.

gamma
[float] Growth index providing an efficient parametrization of the matter perturbations.

Returns

growth_factor
[ndarray, or float if input scalar] The redshift scaling of the growth factor.

References

[1]

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import FlatLambdaCDM
>>> cosmology = FlatLambdaCDM(H0=67.04, Om0=0.3183, Ob0=0.047745)
>>> growth_factor(0, cosmology)
0.5355746155304598
```

skypy.power_spectrum.growth_function

skypy.power_spectrum.**growth_function**(*redshift*, *cosmology*, *gamma=0.5454545454545454*)
Computation of the growth function.

Function used to calculate D(z), growth function at different redshifts, as described in [1] equation 16.

Parameters

redshift
[array_like] Array of redshifts at which to evaluate the growth function.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing methods for the evolution history of omega_matter and omega_lambda with redshift.

gamma
[float] Growth index providing an efficient parametrization of the matter perturbations.

Returns

growth_function
[ndarray] The redshift scaling of the growth function.

References

[1]

Examples

```
>>> import numpy as np
>>> from scipy import integrate
>>> from astropy.cosmology import FlatLambdaCDM
>>> cosmology = FlatLambdaCDM(H0=67.04, Om0=0.3183, Ob0=0.047745)
>>> growth_function(0, cosmology)
0.7909271056297236
```

skypy.power_spectrum.growth_function_carroll

skypy.power_spectrum.**growth_function_carroll**(*redshift*, *cosmology*)
Computation of the growth function.

Return the growth function as a function of redshift for a given cosmology as approximated by Carroll, Press & Turner (1992) Equation 29.

Parameters

redshift
[array_like] Array of redshifts at which to evaluate the growth function.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing methods for the evolution history of omega_matter and omega_lambda with redshift.

Returns

growth
[numpy.ndarray, or float if input scalar] The growth function evaluated at the input redshifts for the given cosmology.

References

doi : 10.1146/annurev.aa.30.090192.002435

Examples

```
>>> import numpy as np
>>> from astropy.cosmology import default_cosmology
>>> redshift = np.array([0, 1, 2])
>>> cosmology = default_cosmology.get()
>>> growth_function_carroll(redshift, cosmology)
array([0.781361..., 0.476280..., 0.327549...])
```

skypy.power_spectrum.growth_function_derivative

skypy.power_spectrum.**growth_function_derivative**(*redshift*, *cosmology*, *gamma=0.5454545454545454*)

Computation of the first derivative of the growth function.

Function used to calculate D'(z), derivative of the growth function with respect to redshift, described in [1] equation 16.

Parameters

redshift
[array_like] Array of redshifts at which to evaluate the growth function.

cosmology
[astropy.cosmology.Cosmology] Cosmology object providing methods for the evolution history of omega_matter and omega_lambda with redshift.

gamma
[float] Growth index providing an efficient parametrization of the matter perturbations.

Returns

growth_function_derivative
[ndarray, or float if input scalar] The redshift scaling of the derivative of the growth function.

References

[1]

Examples

```
>>> import numpy as np
>>> from scipy import integrate
>>> from astropy.cosmology import FlatLambdaCDM
>>> cosmology = FlatLambdaCDM(H0=67.04, Om0=0.3183, Ob0=0.047745)
>>> growth_function_derivative(0, cosmology)
-0.42360048051025856
```

1.4 Utils (skypy.utils)

This module contains utility functions.

1.4.1 Astronomy (skypy.utils.astronomy)

Astronomy utility module.

This module provides methods to convert among astronomical quantities like luminosity and magnitude.

Utility functions

luminosity_from_absolute_magnitude	Converts absolute magnitudes into luminosities
absolute_magnitude_from_luminosity	Converts luminosities into absolute magnitudes

skypy.utils.astronomy.luminosity_from_absolute_magnitude

skypy.utils.astronomy.**luminosity_from_absolute_magnitude**(*absolute_magnitude*)
Converts absolute magnitudes into luminosities

Parameters

absolute_magnitude
[array_like] Input absolute magnitudes

Returns

——-

ndarray, or float if input is scalar

Luminosity values.

skypy.utils.astronomy.absolute_magnitude_from_luminosity

skypy.utils.astronomy.**absolute_magnitude_from_luminosity**(*luminosity*)
Converts luminosities into absolute magnitudes

Parameters

luminosity
[array_like] Input luminosity

Returns

ndarray, or float if input is scalar

Absolute magnitude values

1.4.2 Random sampling (skypy.utils.random)

Random utility module.

This module provides methods to draw from random distributions.

Utility Functions

schechter	Sample from the Schechter function.

skypy.utils.random.schechter

skypy.utils.random.**schechter**(*alpha*, *x_min*, *x_max*, *resolution=100*, *size=None*)
Sample from the Schechter function.

Parameters

alpha
[float or int] The alpha parameter in the Schechter function in [1].

x_min, x_max
[array_like] Lower and upper bounds for the random variable x.

resolution
[int] Resolution of the inverse transform sampling spline. Default is 100.

size: int, optional
Output shape of samples. Default is None.

Returns

x_sample
[array_like] Samples drawn from the Schechter function.

References

[1]

Examples

```
>>> import skypy.utils.random as random
>>> alpha = -1.3
>>> sample = random.schechter(alpha, x_min=1e-10, x_max=1e2,
...                           resolution=100, size=1000)
```

1.4.3 Special functions (skypy.utils.special)

Special functions.

This module computes useful special functions.

Utility functions

upper_incomplete_gamma	Non-regularised upper incomplete gamma function.

skypy.utils.special.upper_incomplete_gamma

skypy.utils.special.**upper_incomplete_gamma**(a, x)

Non-regularised upper incomplete gamma function. Extension of the regularised upper incomplete gamma function implemented in SciPy. In this way you can pass a negative value for a.

Parameters

a
[array_like] Parameter

x
[array_like] Nonnegative parameter

Returns

Scalar or ndarray
Value of the non-regularised upper incomplete gamma function.

CHAPTER

TWO

PIPELINE

2.1 Pipeline (skypy.pipeline)

This module provides methods to pipeline together multiple models with dependencies and handle their outputs.

2.1.1 Driver (skypy.pipeline.driver)

Driver module.

This module provides methods to run pipelines of functions with dependencies and handle their results.

SkyPyDriver	Class for running pipelines.

skypy.pipeline.driver.SkyPyDriver

class skypy.pipeline.driver.**SkyPyDriver**

Class for running pipelines.

This is the main class for running pipelines of functions with dependencies and using their results to generate tables.

__init__(*self*, */*, **args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(self, /, *args, **kwargs)	Initialize self.
execute(self, config[, file_format])	Run a pipeline.

Part III

Developer Documentation

Part IV

Project details

Part V

Index

- genindex
- modindex
- search

Part VI

Acknowledgements

Logo Credit: Maria Fonseca de la Bella

BIBLIOGRAPHY

[1] Herbel J., Kacprzak T., Amara A. et al., 2017, Journal of Cosmology and Astroparticle Physics, Issue 08, article id. 035 (2017)

[1] Kacprzak T., Herbel J., Nicola A. et al., arXiv:1906.01018

[1] Herbel J., Kacprzak T., Amara A. et al., 2017, Journal of Cosmology and Astroparticle Physics, Issue 08, article id. 035 (2017)

[1] Herbel J., Kacprzak T., Amara A. et al., 2017, Journal of Cosmology and Astroparticle Physics, Issue 08, article id. 035 (2017)

[1] Smail I., Ellis R. S., Fitchett M. J., 1994, MNRAS, 270, 245

[2] Amara A., Refregier A., 2007, MNRAS, 381, 1018

[1] D. W. Hogg, (1999), astro-ph/9905116.

[1] Herbel J., Kacprzak T., Amara A. et al., 2017, Journal of Cosmology and Astroparticle Physics, Issue 08, article id. 035 (2017)

[2] Blanton M. R., Roweis S., 2007, The Astronomical Journal, Volume 133, Page 734

[1] Vale A., Ostriker J. P., 2004, MNRAS, 353, 189

[1] Eisenstein D. J., Hu W., ApJ, 496, 605 (1998)

[2] Eisenstein D. J., Hu W., ApJ, 511, 5 (1999)

[3] Komatsu et al., ApJS, 180, 330 (2009)

[1] Eisenstein D. J., Hu W., ApJ, 496, 605 (1998)

[2] Eisenstein D. J., Hu W., ApJ, 511, 5 (1999)

[3] Komatsu et al., ApJS, 180, 330 (2009)

[1] Eisenstein D. J., Hu W., ApJ, 496, 605 (1998)

[2] Eisenstein D. J., Hu W., ApJ, 511, 5 (1999)

[3] Komatsu et al., ApJS, 180, 330 (2009)

[1] R. E. Smith it et al., VIRGO Consortium, Mon. Not. Roy. Astron. Soc. 341, 1311 (2003).

[2] R. Takahashi, M. Sato, T. Nishimichi, A. Taruya and M. Oguri, Astrophys. J. 761, 152 (2012).

[3] S. Bird, M. Viel and M. G. Haehnelt, Mon. Not. Roy. Astron. Soc. 420, 2551 (2012).

[1] R. E. Smith it et al., VIRGO Consortium, Mon. Not. Roy. Astron. Soc. 341, 1311 (2003).

[2] R. Takahashi, M. Sato, T. Nishimichi, A. Taruya and M. Oguri, Astrophys. J. 761, 152 (2012).

[3] S. Bird, M. Viel and M. G. Haehnelt, Mon. Not. Roy. Astron. Soc. 420, 2551 (2012).

[1] R. E. Smith it et al., VIRGO Consortium, Mon. Not. Roy. Astron. Soc. 341, 1311 (2003).

[2] R. Takahashi, M. Sato, T. Nishimichi, A. Taruya and M. Oguri, Astrophys. J. 761, 152 (2012).

[3] S. Bird, M. Viel and M. G. Haehnelt, Mon. Not. Roy. Astron. Soc. 420, 2551 (2012).

[1] R. E. Smith it et al., VIRGO Consortium, Mon. Not. Roy. Astron. Soc. 341, 1311 (2003).

[2] R. Takahashi, M. Sato, T. Nishimichi, A. Taruya and M. Oguri, Astrophys. J. 761, 152 (2012).

[3] S. Bird, M. Viel and M. G. Haehnelt, Mon. Not. Roy. Astron. Soc. 420, 2551 (2012).

[1] E. V. Linder, Phys. Rev. D 72, 043529 (2005)

[1] E. V. Linder, Phys. Rev. D 72, 043529 (2005)

[1] E. V. Linder, Phys. Rev. D 72, 043529 (2005)

[1] https://en.wikipedia.org/wiki/Luminosity_function_(astronomy)

PYTHON MODULE INDEX

S

skypy.galaxy, 7
skypy.galaxy.ellipticity, 9
skypy.galaxy.luminosity, 7
skypy.galaxy.redshift, 10
skypy.galaxy.size, 14
skypy.galaxy.spectrum, 18
skypy.halo, 19
skypy.halo.abundance_matching, 20
skypy.halo.mass, 20
skypy.pipeline, 37
skypy.pipeline.driver, 37
skypy.power_spectrum, 21
skypy.utils, 33
skypy.utils.astronomy, 33
skypy.utils.random, 34
skypy.utils.special, 35

INDEX

Symbols

__init__() (*skypy.pipeline.driver.SkyPyDriver method*), 37
__init__() (*skypy.power_spectrum.HalofitParameters method*), 25

A

absolute_magnitude_from_luminosity() (*in module skypy.utils.astronomy*), 34
angular_size() (*in module skypy.galaxy.size*), 14

B

beta_ellipticity (*in module skypy.galaxy.ellipticity*), 9

C

camb() (*in module skypy.power_spectrum*), 21

D

dirichlet_coefficients() (*in module skypy.galaxy.spectrum*), 18

E

early_type_lognormal() (*in module skypy.galaxy.size*), 15
eisenstein_hu() (*in module skypy.power_spectrum*), 22

G

growth_factor() (*in module skypy.power_spectrum*), 30
growth_function() (*in module skypy.power_spectrum*), 31
growth_function_carroll() (*in module skypy.power_spectrum*), 31
growth_function_derivative() (*in module skypy.power_spectrum*), 32

H

halofit() (*in module skypy.power_spectrum*), 26
halofit_bird() (*in module skypy.power_spectrum*), 29
halofit_smith() (*in module skypy.power_spectrum*), 27
halofit_takahashi() (*in module skypy.power_spectrum*), 28
HalofitParameters (*class in skypy.power_spectrum*), 25
herbel_luminosities() (*in module skypy.galaxy.luminosity*), 7
herbel_pdf() (*in module skypy.galaxy.redshift*), 12
herbel_redshift() (*in module skypy.galaxy.redshift*), 10

L

late_type_lognormal() (*in module skypy.galaxy.size*), 16
linear_lognormal() (*in module skypy.galaxy.size*), 17
luminosity_from_absolute_magnitude() (*in module skypy.utils.astronomy*), 33

S

schechter() (*in module skypy.utils.random*), 34
skypy.galaxy (*module*), 7
skypy.galaxy.ellipticity (*module*), 9
skypy.galaxy.luminosity (*module*), 7
skypy.galaxy.redshift (*module*), 10
skypy.galaxy.size (*module*), 14
skypy.galaxy.spectrum (*module*), 18
skypy.halo (*module*), 19
skypy.halo.abundance_matching (*module*), 20
skypy.halo.mass (*module*), 20
skypy.pipeline (*module*), 37
skypy.pipeline.driver (*module*), 37
skypy.power_spectrum (*module*), 21
skypy.utils (*module*), 33
skypy.utils.astronomy (*module*), 33
skypy.utils.random (*module*), 34
skypy.utils.special (*module*), 35
SkyPyDriver (*class in skypy.pipeline.driver*), 37
smail (*in module skypy.galaxy.redshift*), 13

T

transfer_no_wiggles() (*in module skypy.power_spectrum*), 23
transfer_with_wiggles() (*in module skypy.power_spectrum*), 24

U

upper_incomplete_gamma() (*in module skypy.utils.special*), 35

V

vale_ostriker() (*in module skypy.halo.abundance_matching*), 20