SEYFERT

Release 1.3.7dev

Luca Paganin, Marco Bonici, Stefano Davini

Jun 05, 2021

Contents:

SEYFERT (SurvEY FishEr foRecast Tool) is a code to perform forecast of cosmological parameters measurement for several cosmological probes. Up to now are included:

- Weak Lensing
- Spectroscopic Galaxy Clustering
- Photometric Galaxy Clustering
- Void Clustering

The code is written in Python in a modular way. It is actually interfaced with the Boltzmann solver CAMB, altough other codes can be easily added. The code can be installed executing:

```
./installer.sh
```

1 Cosmology modules

1.1 Cosmology

1.1.1 seyfert.cosmology.cosmology Module

Module hosting the cosmology class and its HDF5 file-interface. The Cosmology possesses cosmological parameters values and it can compute all relevant cosmological functions, like the Hubble parameter and the co-moving distance. For now accepted cosmological parameters are:

- w_0, CPL dark energy equation of state intercept
- w_a, CPL dark energy equation of state slope
- h, the dimensionless Hubble constant
- $\sum m_\nu$, the sum of the neutrino mass eigenstates
- σ_8
- n_s, the scalar spectral index
- Ω_m, the cold matter density parameter
- Ω_b, the baryon density parameter
- Ω_{DE}, the dark energy density parameter

1.1.1.1 Classes

AbstractH5FileIO([root])	
Cosmology([params, flat, model_name, ...])	Class for managing the cosmology.
CosmologyError	
H5Cosmology(**kwargs)	
H5PowerSpectrum(**kwargs)	

continues on next page

Table 1 – continued from previous page

Path(*args, **kwargs)	PurePath subclass that can make system calls.
PhysicalParameter([name, fiducial, ...])	
PowerSpectrum([power_spectrum_config, ...])	Class to manage the matter power spectrum. It can be used to calculate the linear and non linear power spectra,
PowerSpectrumConfig(**kwargs)	

Cosmology

class seyfert.cosmology.cosmology.**Cosmology**(*params=None*, *flat=None*, *model_name=None*, *z_grid=None*, *cosmology_name=None*)

Bases: object

Class for managing the cosmology.

Parameters

- **params** (Optional[Dict[str, *PhysicalParameter*]]) – a dictionary with the cosmological parameters, stored as *PhysicalParameter* instances.
- **z_grid** (Optional[ndarray]) – a np.ndarray storing the redshift grid on which to compute functions.
- **dimensionless_hubble_array** – a np.ndarray storing the values of the dimensionless Hubble parameter $E(z)$ sampled on the grid.
- **dimensionless_comoving_distance_array** – a np.ndarray storing the values of the comoving distance $r(z)$ sampled on the grid.
- **power_spectrum** – the matter power spectrum, instance of PowerSpectrum.

Attributes Summary

E_z	**rtype** ndarray
H0	Hubble constant in $\mathrm{km\,s^{-1}\,Mpc^{-1}}$
H_z	**rtype** ndarray
OmDE	Dark energy density parameter Ω_{DE}
OmK	Curvature density parameter Ω_k
Omb	Baryon density parameter Ω_b
Omm	Cold matter density parameter Ω_m
cosmo_pars_current	Current values of the cosmological parameters
growth_factor_z	The growth factor $D(z)$ as a function of redshift.
h	Dimensionless Hubble constant
k_grid	**rtype** ndarray
mnu	Sum of the neutrino mass eigenstates $\sum m_\nu$ in eV

continues on next page

Table 2 – continued from previous page

ns	Scalar spectral index n_s
r_tilde_z	**rtype** ndarray
r_z	**rtype** ndarray
sigma8	σ_8 parameter
transfer_function_k	**rtype** ndarray
w0	Dark energy w_0 CPL parameter
wa	Dark energy w_a CPL (slope) parameter

Methods Summary

checkParameters()	Method checking values of cosmological parameters, returning error if values are invalid.
computeComovingDistance(z)	Method to compute the comoving distance as a function of the redshift.
computeDimensionlessComovingDistance(z)	Method to compute the dimensionless comoving distance at a given redshift, computed according the formula:
computeDimensionlessHubbleParameter(z)	Method to compute the dimensionless Hubble parameter as a function of the redshift.
computeHubbleParameter(z)	Method to compute the Hubble parameter as a function of the redshift.
computeReciprocalDimensionlessHubbleParameter(z)	Shortcut method to compute integrals of the reciprocal of $E(z)$
computeSigmaR(R)	Method to compute the variance of the filtered matter density fluctuations.
evaluateOverRedshiftGrid()	Method which evaluates the Hubble parameter and the comoving distance on the redshift grid.
evaluatePowerSpectrum(workdir, ...)	Method computing the power spectrum for the given cosmology.
fromHDF5(file[, root, load_power_spectrum])	**rtype** *Cosmology*
loadFromHDF5(file[, root, load_power_spectrum])	**rtype** None
saveToHDF5(file[, root, save_power_spectrum])	**rtype** None
sigmaR(R, P_lin, k)	Function to compute the value σ_R of the matter fluctuation variance filtered at a given radius value R

Attributes Documentation

E_z

Return type ndarray

H0

Hubble constant in $\mathrm{km\,s^{-1}\,Mpc^{-1}}$

Return type float

H_z

Return type ndarray

OmDE

Dark energy density parameter Ω_{DE}

Return type float

OmK

Curvature density parameter Ω_k

Return type float

Omb

Baryon density parameter Ω_b

Return type float

Omm

Cold matter density parameter Ω_m

Return type float

cosmo_pars_current

Current values of the cosmological parameters

Return type Dict[str, float]

growth_factor_z

The growth factor $D(z)$ as a function of redshift. The convention is that this growth factor is a decreasing function of z.

Return type ndarray

h

Dimensionless Hubble constant

Return type float

k_grid

Return type ndarray

mnu

Sum of the neutrino mass eigenstates $\sum m_\nu$ in eV

Return type float

ns

Scalar spectral index n_s

Return type float

r_tilde_z

Return type ndarray

r_z

Return type ndarray

sigma8
σ_8 parameter

Return type float

transfer_function_k

Return type ndarray

w0
Dark energy w_0 CPL parameter

Return type float

wa
Dark energy w_a CPL (slope) parameter

Return type float

Methods Documentation

checkParameters()
Method checking values of cosmological parameters, returning error if values are invalid.

Return type None

computeComovingDistance(*z*)
Method to compute the comoving distance as a function of the redshift.

Return type Union[float, ndarray]

computeDimensionlessComovingDistance(*z*)
Method to compute the dimensionless comoving distance at a given redshift, computed according the formula:

$$\tilde{r}(z) = \int_0^z \frac{dz}{E(z)}$$

Parameters **z** (Union[float, ndarray]) – the value of the redshift at which to compute the comoving distance

Return type Union[float, ndarray]

Returns the value of the comoving distance at the given redshift(s) z

computeDimensionlessHubbleParameter(*z*)
Method to compute the dimensionless Hubble parameter as a function of the redshift.

Return type Union[float, ndarray]

computeHubbleParameter(*z*)
Method to compute the Hubble parameter as a function of the redshift.

Return type Union[float, ndarray]

computeReciprocalDimensionlessHubbleParameter(*z*)
Shortcut method to compute integrals of the reciprocal of $E(z)$

Return type Union[float, ndarray]

computeSigmaR(*R*)
Method to compute the variance of the filtered matter density fluctuations. It computes the variance using a top-hat filter in Fourier space with at a given radius R

Parameters **R** (Union[int, float, ndarray]) – radius or radii grid, units are Mpc

Return type Union[float, ndarray]

Returns the value(s) of σ_R at given radius value(s)

evaluateOverRedshiftGrid()
Method which evaluates the Hubble parameter and the comoving distance on the redshift grid.

Return type None

evaluatePowerSpectrum(*workdir*, *power_spectrum_config*)
Method computing the power spectrum for the given cosmology. It will make a call to the Boltzmann code selected for the computation.

Parameters

- **workdir** (Union[str, Path]) – working directory
- **power_spectrum_config** (PowerSpectrumConfig) – power spectrum configuration, instance of PowerSpectrumConfig.

Return type None

classmethod fromHDF5(*file*, *root='/'*, *load_power_spectrum=True*)

Return type *Cosmology*

loadFromHDF5(*file*, *root='/'*, *load_power_spectrum=True*)

Return type None

saveToHDF5(*file*, *root='/'*, *save_power_spectrum=True*)

Return type None

static sigmaR(*R*, *P_lin*, *k*)
Function to compute the value σ_R of the matter fluctuation variance filtered at a given radius value R

Parameters

- **R** (Union[int, float]) – the radius of the filter
- **P_lin** (ndarray) – the linear matter power spectrum as a function of the wavenumber k
- **k** (ndarray) – the wavenumber grid k

Return type float

Returns the value of σ_R

CosmologyError

exception seyfert.cosmology.cosmology.**CosmologyError**

H5Cosmology

class seyfert.cosmology.cosmology.**H5Cosmology**(***kwargs*)
Bases: seyfert.file_io.hdf5_io.AbstractH5FileIO

Methods Summary

readBuildingData()	**rtype** None
writeObjectToFile(obj)	**rtype** None
writeToObject(obj)	**rtype** None

Methods Documentation

readBuildingData()

Return type None

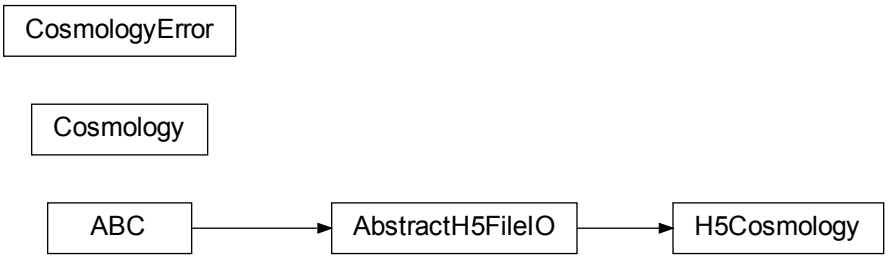
writeObjectToFile(*obj*)

Return type None

writeToObject(*obj*)

Return type None

1.1.1.2 Class Inheritance Diagram



1.2 Physical parameters

1.2.1 seyfert.cosmology.parameter Module

1.2.1.1 Classes

GenericDictInterface([data_dict])	
ParameterError	
Path(*args, **kwargs)	PurePath subclass that can make system calls.
PhysParJSONEncoder(*[, skipkeys, ...])	
PhysicalParameter([name, fiducial, ...])	
PhysicalParametersCollection([...])	

ParameterError

exception seyfert.cosmology.parameter.**ParameterError**

PhysicalParameter

class seyfert.cosmology.parameter.**PhysicalParameter**(*name=None*, *fiducial=None*, *current_value=None*, *kind=None*, *probe=None*, *is_free_parameter=None*, *stem_factor=1.0*, *description=''*, *units=None*, *derivative_method='SteM'*)

Bases: `object`

Attributes Summary

COSMO_PAR_STRING	
NUISANCE_PAR_STRING	
is_cosmological	**rtype** bool
is_galaxy_bias_parameter	**rtype** bool
is_nuisance	**rtype** bool

Methods Summary

computeSteMValues(stem_eps_arr)	**rtype** ndarray
computeValueForDisplacement(eps)	**rtype** float
createCosmologicalParameter(**kwargs)	**rtype** *PhysicalParameter*
createNuisanceParameter(**kwargs)	**rtype** *PhysicalParameter*
fromJSON(json_file)	**rtype** *PhysicalParameter*
from_dict(data)	**rtype** *PhysicalParameter*
resetCurrentValueToFiducial()	

continues on next page

Table 7 – continued from previous page

to_JSON()	**rtype** str
to_dict()	**rtype** Dict[str, Union[str, float, bool]]
updateValueForDisplacement(eps)	

Attributes Documentation

COSMO_PAR_STRING = 'CosmologicalParameter'

NUISANCE_PAR_STRING = 'NuisanceParameter'

is_cosmological

Return type bool

is_galaxy_bias_parameter

Return type bool

is_nuisance

Return type bool

Methods Documentation

computeSteMValues(*stem_eps_arr*)

Return type ndarray

computeValueForDisplacement(*eps*)

Return type float

classmethod createCosmologicalParameter(***kwargs*)

Return type *PhysicalParameter*

classmethod createNuisanceParameter(***kwargs*)

Return type *PhysicalParameter*

classmethod fromJSON(*json_file*)

Return type *PhysicalParameter*

classmethod from_dict(*data*)

Return type *PhysicalParameter*

resetCurrentValueToFiducial()

to_JSON()

Return type str

to_dict()

Return type Dict[str, Union[str, float, bool]]

updateValueForDisplacement(*eps*)

PhysicalParametersCollection

class seyfert.cosmology.parameter.**PhysicalParametersCollection**(*base_stem_disps=None*, *is_universe_flat=None*, ***kwargs*)

Bases: seyfert.base_structs.generic_dict.GenericDictInterface[str, *seyfert.cosmology.parameter.PhysicalParameter*]

Attributes Summary

cosmo_pars_current_values	**rtype** Dict[str, float]
cosmo_pars_fiducials	**rtype** Dict[str, float]
cosmological_parameters	**rtype** Dict[str, *PhysicalParameter*]
free_cosmo_pars_fiducials	**rtype** Dict[str, float]
free_cosmological_parameters	**rtype** Dict[str, *PhysicalParameter*]
free_physical_parameters	**rtype** Dict[str, *PhysicalParameter*]
nuisance_parameters	**rtype** Dict[str, *PhysicalParameter*]
params	**rtype** Dict[str, *PhysicalParameter*]

Methods Summary

computePhysParSTEMValues([dvar])	**rtype** ndarray
fromJSON(file[, only_cosmological])	**rtype** *PhysicalParametersCollection*
from_dict_list(dict_list[, only_cosmological])	**rtype** *PhysicalParametersCollection*
getFreeNuisanceParametersForProbe(probe)	**rtype** Dict[str, *PhysicalParameter*]
getNuisanceParametersForProbe(probe)	**rtype** Dict[str, *PhysicalParameter*]
getParamsDictFromDictList(dict_list[, ...])	**rtype** Dict[str, *PhysicalParameter*]
loadStemDisplacements(base_stem_disp)	**rtype** None
readJSON(file[, only_cosmological])	**rtype** None
resetPhysicalParametersToFiducial()	**rtype** None
updatePhysicalParametersForDvarStep([dvar, step])	**rtype** None
writeJSON(file)	**rtype** None

Attributes Documentation

cosmo_pars_current_values

Return type Dict[str, float]

cosmo_pars_fiducials

Return type Dict[str, float]

cosmological_parameters

Return type Dict[str, *PhysicalParameter*]

free_cosmo_pars_fiducials

Return type Dict[str, float]

free_cosmological_parameters

Return type Dict[str, *PhysicalParameter*]

free_physical_parameters

Return type Dict[str, *PhysicalParameter*]

nuisance_parameters

Return type Dict[str, *PhysicalParameter*]

params

Return type Dict[str, *PhysicalParameter*]

Methods Documentation

computePhysParSTEMValues(*dvar=None*)

Return type ndarray

classmethod fromJSON(*file*, *only_cosmological=False*)

Return type *PhysicalParametersCollection*

classmethod from_dict_list(*dict_list*, *only_cosmological=False*)

Return type *PhysicalParametersCollection*

getFreeNuisanceParametersForProbe(*probe*)

Return type Dict[str, *PhysicalParameter*]

getNuisanceParametersForProbe(*probe*)

Return type Dict[str, *PhysicalParameter*]

static getParamsDictFromDictList(*dict_list*, *only_cosmological=False*)

Return type Dict[str, *PhysicalParameter*]

loadStemDisplacements(*base_stem_disp*)

Return type None

readJSON(*file*, *only_cosmological=False*)

Return type None

resetPhysicalParametersToFiducial()

Return type None

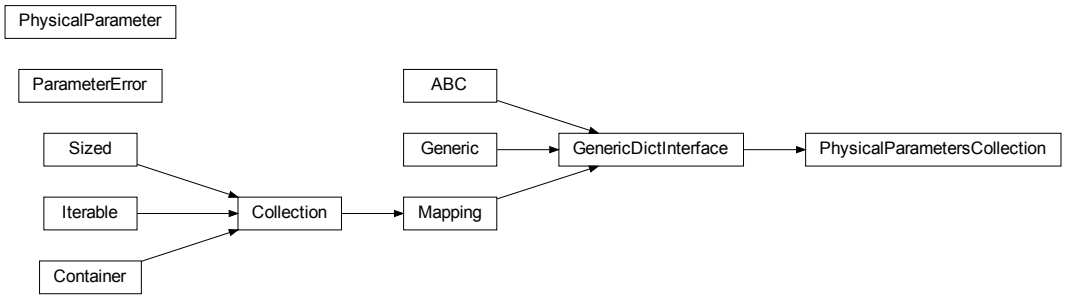
updatePhysicalParametersForDvarStep(*dvar=None*, *step=None*)

Return type None

writeJSON(*file*)

Return type None

1.2.1.2 Class Inheritance Diagram



1.3 Power Spectrum

1.3.1 seyfert.cosmology.power_spectrum Module

Module hosting PowerSpectrum class and H5PowerSpectrum, its HDF5 file-interface class.

1.3.1.1 Classes

AbstractH5FileIO([root])	
H5PowerSpectrum(**kwargs)	
Path(*args, **kwargs)	PurePath subclass that can make system calls.
PhysicalParameter([name, fiducial, ...])	
PowerSpectrum([power_spectrum_config, ...])	Class to manage the matter power spectrum. It can be used to calculate the linear and non linear power spectra,
PowerSpectrumConfig(**kwargs)	

H5PowerSpectrum

class seyfert.cosmology.power_spectrum.**H5PowerSpectrum**(***kwargs*)

Bases: seyfert.file_io.hdf5_io.AbstractH5FileIO

Methods Summary

writeObjectToFile(obj)	**rtype** None
writeToObject(obj)	**rtype** None

Methods Documentation

writeObjectToFile(*obj*)

Return type None

writeToObject(*obj*)

Return type None

PowerSpectrum

class seyfert.cosmology.power_spectrum.**PowerSpectrum**(*power_spectrum_config=None*, *cosmo_pars=None*)

Bases: object

Class to manage the matter power spectrum. It can be used to calculate the linear and non linear power spectra, using a given boltzmann code (e.g. CAMB), or as a container storing all the relevant information about the matter power spectrum. In order to compute the power spectra an instance the class ExternalBoltzmannSolver is used. Methods for doing file I/O are present, and the file format employed is HDF5 (.h5), since it allows to store binary data with high compression options.

Parameters

- **k_grid** – the wavenumber grid over which the power spectra are evaluated
- **z_grid** – the redshift grid over which the power spectra are evaluated
- **lin_p_mm_z_k** – the linear matter power spectrum
- **nonlin_p_mm_z_k** – the nonlinear matter power spectrum

Attributes Summary

growth_factor_z	Returns the growth factor from the linear matter power spectrum as D(z) = Plin(z, k_min) / Plin(0, k_min).

Methods Summary

evaluateLinearAndNonLinearPowerSpectra(workdir)	Method for calling the selected Boltzmann solver.
fromHDF5(file[, root])	**rtype** *PowerSpectrum*
getResultsFromMatterPowerGenerator()	Method for read and store into attributes the Boltzmann solver results.
loadFromHDF5(file[, root])	**rtype** None
saveToHDF5(file[, root])	**rtype** None

Attributes Documentation

growth_factor_z
Returns the growth factor from the linear matter power spectrum as D(z) = Plin(z, k_min) / Plin(0, k_min). The scale dependency of the growth is not taken into account. :rtype: ndarray :return: np.ndarray

Methods Documentation

evaluateLinearAndNonLinearPowerSpectra(*workdir*)
Method for calling the selected Boltzmann solver.

Return type None

classmethod fromHDF5(*file*, *root='/'*)

Return type *PowerSpectrum*

getResultsFromMatterPowerGenerator()
Method for read and store into attributes the Boltzmann solver results.

Return type None

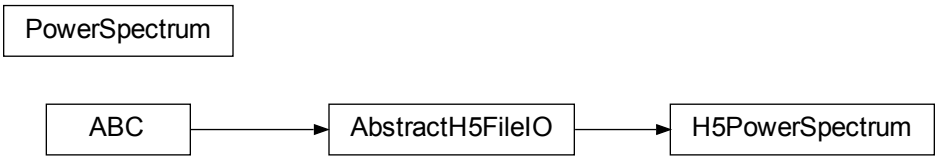
loadFromHDF5(*file*, *root='/'*)

Return type None

saveToHDF5(*file*, *root='/'*)

Return type None

1.3.1.2 Class Inheritance Diagram



1.4 Boltzmann solver

1.4.1 seyfert.cosmology.boltzmann_solver Module

1.4.1.1 Classes

ABC()	Helper class that provides a standard way to create an ABC using inheritance.
CAMBBoltzmannSolver(*args)	
CLASSBoltzmannSolver(*args)	
ExternalBoltzmannSolver(workdir, config, ...)	
Path(*args, **kwargs)	PurePath subclass that can make system calls.
PhysicalParameter([name, fiducial, ...])	
PowerSpectrumConfig(**kwargs)	

CAMBBoltzmannSolver

class seyfert.cosmology.boltzmann_solver.**CAMBBoltzmannSolver**(**args*)

Bases: *seyfert.cosmology.boltzmann_solver.ExternalBoltzmannSolver*

Methods Summary

computeCAMBCosmologicalBasis()	**rtype** Dict
computeCAMBRedshiftGroupsNumber(n_redshifts)	**rtype** int
evaluateLinearAndNonLinearPowerSpectra()	
finetuneScalarAmpFromSigma8()	**rtype** None
getTransferFunctionFromResults(camb_results)	**rtype** ndarray
readCAMBIniFileToDict(ini_file)	**rtype** Dict[str, str]
run([get_growth, get_transfer_func])	
writeDictToIniFile(ini_dict, file)	**rtype** None
writeUpdatedCAMBIniFile()	

Methods Documentation

computeCAMBCosmologicalBasis()

Return type Dict

static computeCAMBRedshiftGroupsNumber(*n_redshifts*)

Return type int

evaluateLinearAndNonLinearPowerSpectra()

finetuneScalarAmpFromSigma8()

Return type None

static getTransferFunctionFromResults(*camb_results*)

Return type ndarray

static readCAMBIniFileToDict(*ini_file*)

Return type Dict[str, str]

run(*get_growth=True*, *get_transfer_func=True*)

static writeDictToIniFile(*ini_dict*, *file*)

Return type None

writeUpdatedCAMBIniFile()

CLASSBoltzmannSolver

class seyfert.cosmology.boltzmann_solver.**CLASSBoltzmannSolver**(**args*)

Bases: *seyfert.cosmology.boltzmann_solver.ExternalBoltzmannSolver*

Methods Summary

computeCLASSCosmologicalBasis()	**rtype** Dict
evaluateLinearAndNonLinearPowerSpectra()	
run()	

Methods Documentation

computeCLASSCosmologicalBasis()

Return type Dict

evaluateLinearAndNonLinearPowerSpectra()

run()

ExternalBoltzmannSolver

class seyfert.cosmology.boltzmann_solver.**ExternalBoltzmannSolver**(*workdir*, *config*, *cosmological_parameters*)

Bases: abc.ABC

Methods Summary

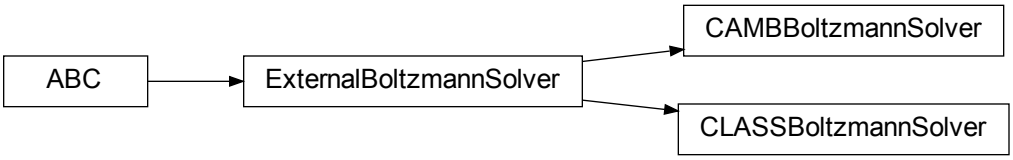
evaluateLinearAndNonLinearPowerSpectra()	
run()	

Methods Documentation

abstract evaluateLinearAndNonLinearPowerSpectra()

abstract run()

1.4.1.2 Class Inheritance Diagram



1.5 Redshift Density

1.5.1 seyfert.cosmology.redshift_density Module

1.5.1.1 Functions

compute_photoXspectro_shotnoise_ij(nph, nsp)	**rtype** ndarray
get_bins_overlap(z1_l, z1_r, z2_l, z2_r)	**rtype** Tuple[float, float]
jit([signature_or_function, locals, target, ...])	This decorator is used to compile a Python function into native code.

compute_photoXspectro_shotnoise_ij

seyfert.cosmology.redshift_density.**compute_photoXspectro_shotnoise_ij**(*nph*, *nsp*)

Return type ndarray

get_bins_overlap

seyfert.cosmology.redshift_density.**get_bins_overlap**(*z1_l*, *z1_r*, *z2_l*, *z2_r*)

Return type Tuple[float, float]

1.5.1.2 Classes

AbstractH5FileIO([root])	
DensityError	
H5Density(**kwargs)	
Path(*args, **kwargs)	PurePath subclass that can make system calls.
RedshiftDensity([probe])	
TypingError(msg[, loc, highlighting])	A type inference failure.

DensityError

exception seyfert.cosmology.redshift_density.**DensityError**

H5Density

class seyfert.cosmology.redshift_density.**H5Density**(**kwargs*)

Bases: seyfert.file_io.hdf5_io.AbstractH5FileIO

Methods Summary

writeObjectToFile(obj)	**rtype** None
writeToObject(obj)	**rtype** None

Methods Documentation

writeObjectToFile(*obj*)

Return type None

writeToObject(*obj*)

Return type None

RedshiftDensity

class seyfert.cosmology.redshift_density.**RedshiftDensity**(*probe=None*)
Bases: object

Attributes Summary

n_bins	**rtype** int
shot_noise	**rtype** ndarray
z_bin_centers	**rtype** ndarray
z_max	**rtype** float
z_min	**rtype** float

Methods Summary

computeBinNormFactor(bin_idx)	**rtype** float
computeInstrumentResponse(z_p, z)	**rtype** Union[float, ndarray]
computeNormalizedDensityAtBinAndRedshift(i, z)	**rtype** Union[float, ndarray]
computeSurfaceDensityAtBin(i)	**rtype** float

continues on next page

Table 22 – continued from previous page

computeTotalGalaxyNumber([integ_method])	**rtype** float
convolvedNdzdOmegaWithInstrumentResponse(z, i)	**rtype** Union[float, ndarray]
evaluate(z_grid)	**rtype** None
evaluateBinNormFactors()	**rtype** None
evaluateSurfaceDensity()	
fromHDF5(file[, root])	**rtype** *RedshiftDensity*
interpolateInput()	**rtype** None
loadFromHDF5(file[, root])	**rtype** None
modifiedGaussianResponse(z_p, z[, z_mean, ...])	**rtype** Union[float, ndarray]
saveToHDF5(file[, root])	**rtype** None
setUp()	**rtype** None

Attributes Documentation

n_bins

Return type int

shot_noise

Return type ndarray

z_bin_centers

Return type ndarray

z_max

Return type float

z_min

Return type float

Methods Documentation

computeBinNormFactor(*bin_idx*)

Return type float

computeInstrumentResponse(*z_p*, *z*)

Return type Union[float, ndarray]

computeNormalizedDensityAtBinAndRedshift(*i*, *z*)

Return type Union[float, ndarray]

computeSurfaceDensityAtBin(*i*)

Return type float

computeTotalGalaxyNumber(*integ_method='simps'*)

Return type float

convolvedNdzdOmegaWithInstrumentResponse(*z*, *i*)

Return type Union[float, ndarray]

evaluate(*z_grid*)

Return type None

evaluateBinNormFactors()

Return type None

evaluateSurfaceDensity()

static fromHDF5(*file*, *root='/'*)

Return type *RedshiftDensity*

interpolateInput()

Return type None

loadFromHDF5(*file*, *root='/'*)

Return type None

static modifiedGaussianResponse(*z_p*, *z*, *z_mean=None*, *sigma=None*, *c=None*, *amplitude=None*)

Return type Union[float, ndarray]

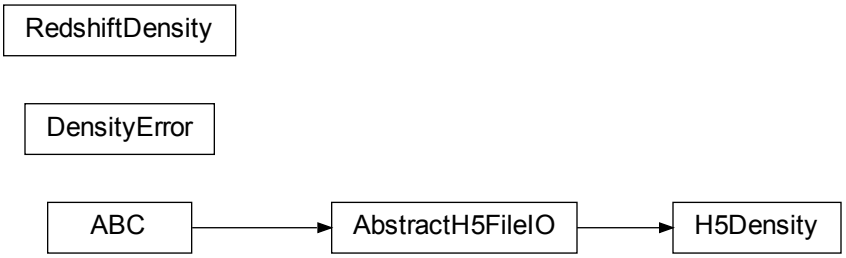
saveToHDF5(*file*, *root='/'*)

Return type None

`setUp()`

Return type None

1.5.1.3 Class Inheritance Diagram



1.6 Bias

1.6.1 seyfert.cosmology.bias Module

Module hosting the bias class and the implemented bias models.

1.6.1.1 Classes

`ABC`()	Helper class that provides a standard way to create an ABC using inheritance.
`AbstractH5FileIO`([root])	
Bias([probe, z_bin_edges])	Class managing the bias.
BiasError	
BiasModel([name, z_bin_edges, cosmology, ...])	
ConstantBias(**kwargs)	
EuclidFlagshipGCphBias(**kwargs)	
FiducialGrowthVoidBias(**kwargs)	
H5Bias(**kwargs)	

continues on next page

Table 23 – continued from previous page

Path(*args, **kwargs)	PurePath subclass that can make system calls.
PiecewiseBias(**kwargs)	
VdnVoidBias(**kwargs)	

Bias

class seyfert.cosmology.bias.**Bias**(*probe=None*, *z_bin_edges=None*)

Bases: object

Class managing the bias.

Parameters

- **model** – instance of *BiasModel*. It defines the model with which to compute the bias as a function of the redshift.
- **nuisance_parameters** – a Dict[str, float] of nuisance parameters for the bias. It should contain the updated values of the nuisance parameters, since these may be let free to vary.
- **additional_parameters** – a Dict[str, float] of additional parameters for the bias. These are kept constant and not let free to vary as it happens for nuisance parameters

Attributes Summary

has_output	**rtype** bool

Methods Summary

evaluateBias(z_grid)	**rtype** None
fromHDF5(file[, root])	**rtype** *Bias*
initBiasModel(**kwargs)	Method for initializing the bias model.
loadFromHDF5(file[, root])	**rtype** None
saveToHDF5(file[, root])	**rtype** None

Attributes Documentation

has_output

Return type bool

Methods Documentation

evaluateBias(*z_grid*)

Return type None

static fromHDF5(*file*, *root='/'*)

Return type *Bias*

initBiasModel(***kwargs*)
Method for initializing the bias model.

Parameters **kwargs** – generic keyword arguments parameters. These depend on the particular model that is being instantiated.

Return type None

loadFromHDF5(*file*, *root='/'*)

Return type None

saveToHDF5(*file*, *root='/'*)

Return type None

BiasError

exception seyfert.cosmology.bias.**BiasError**

BiasModel

class seyfert.cosmology.bias.**BiasModel**(*name=None*, *z_bin_edges=None*, *cosmology=None*, *nuisance_parameters=None*, *additional_parameters=None*)

Bases: abc.ABC

Attributes Summary

n_bins	**rtype** int
z_bin_centers	**rtype** ndarray

Methods Summary

computeBias(z_grid)	**rtype** ndarray

Attributes Documentation

n_bins

Return type int

z_bin_centers

Return type ndarray

Methods Documentation

abstract computeBias(*z_grid*)

Return type ndarray

ConstantBias

class seyfert.cosmology.bias.**ConstantBias**(***kwargs*)

Bases: *seyfert.cosmology.bias.BiasModel*

Methods Summary

computeBias(z_grid)	**rtype** ndarray

Methods Documentation

computeBias(*z_grid*)

Return type ndarray

EuclidFlagshipGCphBias

class seyfert.cosmology.bias.**EuclidFlagshipGCphBias**(***kwargs*)

Bases: *seyfert.cosmology.bias.BiasModel*

Attributes Summary

Aph	**rtype** float
Bph	**rtype** float
Cph	**rtype** float
Dph	**rtype** float

Methods Summary

computeBias(z_grid)	**rtype** ndarray

Attributes Documentation

Aph

Return type float

Bph

Return type float

Cph

Return type float

Dph

Return type float

Methods Documentation

computeBias(*z_grid*)

Return type ndarray

FiducialGrowthVoidBias

class seyfert.cosmology.bias.**FiducialGrowthVoidBias**(***kwargs*)
Bases: *seyfert.cosmology.bias.BiasModel*

Methods Summary

computeBias(z_grid)	**rtype** ndarray

Methods Documentation

computeBias(*z_grid*)

Return type ndarray

H5Bias

class seyfert.cosmology.bias.**H5Bias**(***kwargs*)
Bases: seyfert.file_io.hdf5_io.AbstractH5FileIO

Methods Summary

writeObjectToFile(obj)	**rtype** None
writeToObject(obj)	**rtype** None

Methods Documentation

writeObjectToFile(*obj*)

Return type None

writeToObject(*obj*)

Return type None

PiecewiseBias

class seyfert.cosmology.bias.**PiecewiseBias**(***kwargs*)
Bases: *seyfert.cosmology.bias.BiasModel*

Methods Summary

computeBias(z_grid)	**rtype** ndarray
getSortedBiasValues()	**rtype** ndarray

Methods Documentation

computeBias(*z_grid*)

Return type ndarray

getSortedBiasValues()

Return type ndarray

VdnVoidBias

class seyfert.cosmology.bias.**VdnVoidBias**(***kwargs*)

Bases: *seyfert.cosmology.bias.BiasModel*

Attributes Summary

R_max_Mpc	**rtype** float
R_min_Mpc	**rtype** float
delta_c0	**rtype** float
delta_v0	**rtype** float
n_R	**rtype** int
z_grid	

Methods Summary

computeBias(z_grid)	**rtype** ndarray
computeVdnSizeFunction(R, D, x, delta_v_z, ...)	
computeVoidMultiplicity(D, x, delta_v_z, sigmaR)	**rtype** ndarray
voidMultiplicity(delta_v, D, sigma, x)	**rtype** float

Attributes Documentation

R_max_Mpc

Return type float

R_min_Mpc

Return type float

delta_c0

Return type float

delta_v0

Return type float

n_R

Return type int

z_grid

Methods Documentation

computeBias(*z_grid*)

Return type ndarray

computeVdnSizeFunction(*R*, *D*, *x*, *delta_v_z*, *sigmaR*)

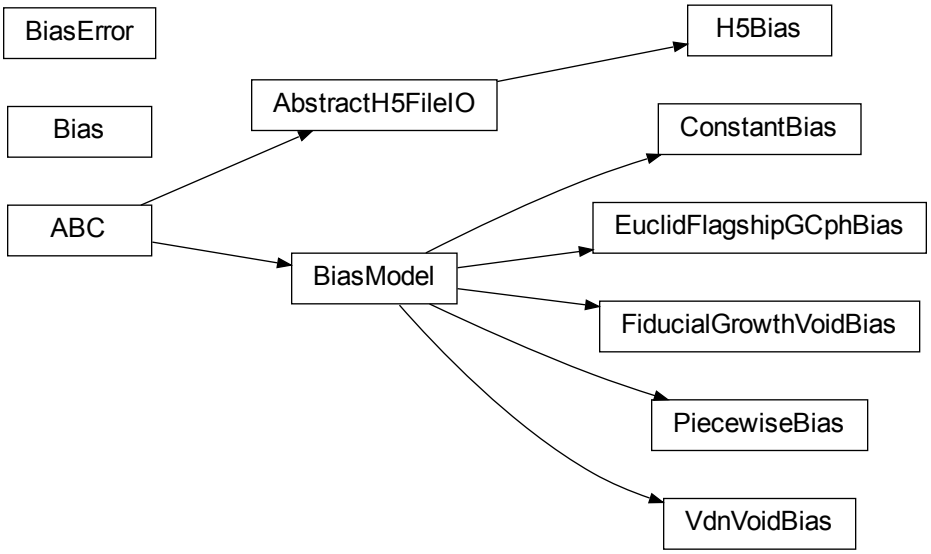
computeVoidMultiplicity(*D*, *x*, *delta_v_z*, *sigmaR*)

Return type ndarray

static voidMultiplicity(*delta_v*, *D*, *sigma*, *x*)

Return type float

1.6.1.2 Class Inheritance Diagram



1.7 Weight functions

1.7.1 seyfert.cosmology.weight_functions Module

1.7.1.1 Functions

abstractmethod(funcobj)	A decorator indicating abstract methods.
join(a, *p)	Join two or more pathname components, inserting '/' as needed.
probe_from_weight_function_cls_name(...)	
weight_cls_name_from_obs(obs_string)	
weight_function_for_probe(probe[, ...])	**rtype** *WeightFunction*

probe_from_weight_function_cls_name

seyfert.cosmology.weight_functions.**probe_from_weight_function_cls_name**(*weight_function_name*)

weight_cls_name_from_obs

seyfert.cosmology.weight_functions.**weight_cls_name_from_obs**(*obs_string*)

weight_function_for_probe

seyfert.cosmology.weight_functions.**weight_function_for_probe**(*probe*, *probe_config=None*, *nuisance_params=None*, *cosmology=None*, *fiducial_cosmology=None*, *angular_config=None*)

Return type *WeightFunction*

1.7.1.2 Classes

ABC()	Helper class that provides a standard way to create an ABC using inheritance.
AbstractH5FileIO([root])	
AngularConfig(**kwargs)	
Bias([probe, z_bin_edges])	Class managing the bias.
Cosmology([params, flat, model_name, ...])	Class for managing the cosmology.

continues on next page

Table 37 – continued from previous page

GalaxyClusteringWeightFunction(**kwargs)	
H5WeightFunction(**kwargs)	
LensingWeightFunction(**kwargs)	
Path(*args, **kwargs)	PurePath subclass that can make system calls.
PhotometricGalaxyWeightFunction(**kwargs)	
PhysicalParameter([name, fiducial, ...])	
ProbeConfig(name[, input_data_dir, fc_xml_io])	
RedshiftDensity([probe])	
SpectroscopicGalaxyWeightFunction(**kwargs)	
VoidWeightFunction(**kwargs)	
WeightFunction([probe_config, ...])	
WeightFunctionWithBias(**kwargs)	

GalaxyClusteringWeightFunction

class seyfert.cosmology.weight_functions.**GalaxyClusteringWeightFunction**(***kwargs*)

Bases: *seyfert.cosmology.weight_functions.WeightFunctionWithBias*, abc.ABC

Methods Summary

setupBias()	**rtype** None

Methods Documentation

setupBias()

Return type None

H5WeightFunction

class seyfert.cosmology.weight_functions.**H5WeightFunction**(***kwargs*)
Bases: seyfert.file_io.hdf5_io.AbstractH5FileIO

Methods Summary

readBuildingData()	**rtype** None
writeObjectToFile(obj)	**rtype** None
writeToObject(obj)	**rtype** None

Methods Documentation

readBuildingData()

Return type None

writeObjectToFile(*obj*)

Return type None

writeToObject(*obj*)

Return type None

LensingWeightFunction

class seyfert.cosmology.weight_functions.**LensingWeightFunction**(***kwargs*)
Bases: *seyfert.cosmology.weight_functions.WeightFunction*

Attributes Summary

H0	**rtype** float
Omm	**rtype** float
c_km_s	**rtype** float

Methods Summary

computeIntrinsicAlignmentContribution()	**rtype** ndarray
computeLensingEfficiency(z)	**rtype** ndarray
computeLensingEfficiencyAtBin(i, z)	**rtype** Union[float, ndarray]
evaluateOverRedshiftGrid(z_grid)	**rtype** None

Attributes Documentation

H0

Return type float

Omm

Return type float

c_km_s

Return type float

Methods Documentation

computeIntrinsicAlignmentContribution()

Return type ndarray

computeLensingEfficiency(*z*)

Return type ndarray

computeLensingEfficiencyAtBin(*i*, *z*)

Return type Union[float, ndarray]

evaluateOverRedshiftGrid(*z_grid*)

Return type None

PhotometricGalaxyWeightFunction

class seyfert.cosmology.weight_functions.**PhotometricGalaxyWeightFunction**(***kwargs*)
Bases: *seyfert.cosmology.weight_functions.GalaxyClusteringWeightFunction*

Methods Summary

setupBias()	**rtype** None

Methods Documentation

setupBias()

Return type None

SpectroscopicGalaxyWeightFunction

class seyfert.cosmology.weight_functions.**SpectroscopicGalaxyWeightFunction**(***kwargs*)
Bases: *seyfert.cosmology.weight_functions.GalaxyClusteringWeightFunction*

Methods Summary

setupBias()	**rtype** None

Methods Documentation

setupBias()

Return type None

VoidWeightFunction

class seyfert.cosmology.weight_functions.**VoidWeightFunction**(***kwargs*)

Bases: *seyfert.cosmology.weight_functions.WeightFunctionWithBias*

Methods Summary

setupBias()	**rtype** None

Methods Documentation

setupBias()

Return type None

WeightFunction

class seyfert.cosmology.weight_functions.**WeightFunction**(*probe_config=None*, *nuisance_params=None*, *cosmology=None*, *fiducial_cosmology=None*, *angular_config=None*)

Bases: abc.ABC

Attributes Summary

H_z	**rtype** ndarray
is_evaluated	**rtype** bool
n_bins	
n_i_z	**rtype** ndarray
probe	
r_tilde_z	**rtype** ndarray
z_bin_centers	
z_bin_edges	

continues on next page

Table 45 – continued from previous page

z_max
z_min

Methods Summary

evaluateOverRedshiftGrid(z_grid)
fromHDF5(file[, root]) **rtype** *WeightFunction*
loadFromHDF5(file[, root]) **rtype** None
saveToHDF5(file[, root]) **rtype** None
setUp()

Attributes Documentation

H_z

Return type ndarray

is_evaluated

Return type bool

n_bins

n_i_z

Return type ndarray

probe

r_tilde_z

Return type ndarray

z_bin_centers

z_bin_edges

z_max

z_min

Methods Documentation

abstract evaluateOverRedshiftGrid(*z_grid*)

classmethod fromHDF5(*file*, *root='/'*)

Return type *WeightFunction*

loadFromHDF5(*file*, *root='/'*)

Return type None

saveToHDF5(*file*, *root='/'*)

Return type None

setUp()

WeightFunctionWithBias

class seyfert.cosmology.weight_functions.**WeightFunctionWithBias**(***kwargs*)

Bases: *seyfert.cosmology.weight_functions.WeightFunction*, abc.ABC

Methods Summary

evaluateOverRedshiftGrid(z_grid)	**rtype** None
setUp()	
setupBias()	**rtype** None

Methods Documentation

evaluateOverRedshiftGrid(*z_grid*)

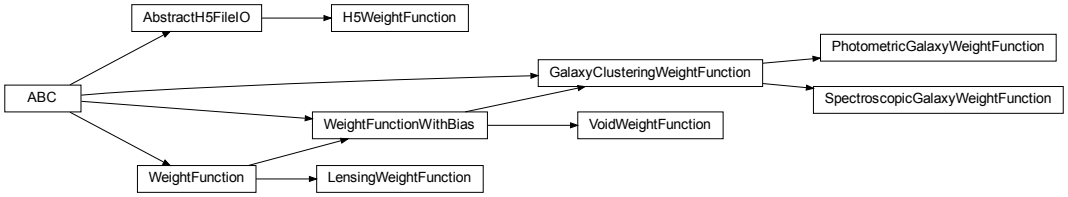
Return type None

setUp()

abstract setupBias()

Return type None

1.7.1.3 Class Inheritance Diagram



1.8 Angular power spectra

1.8.1 seyfert.cosmology.c_ells Module

Module hosting the classes for computing and storing the angular power spectra.

1.8.1.1 Functions

cl_key_long_to_short(cl_long_key)	**rtype** str
cl_key_short_to_long(cl_short_key)	**rtype** str
grid_plot(x, y[, label, axes, idx_fontsize, ...])	
join(a, *p)	Join two or more pathname components, inserting '/' as needed.

cl_key_long_to_short

seyfert.cosmology.c_ells.**cl_key_long_to_short**(*cl_long_key*)

Return type str

cl_key_short_to_long

seyfert.cosmology.c_ells.**cl_key_short_to_long**(*cl_short_key*)

Return type str

1.8.1.2 Classes

AbstractH5FileIO([root])	
AngularCoefficient([probe1, probe2, kernel, ...])	Class representing a particular kind of angular power spectrum (i.e.
AngularCoefficientsCollector([phys_params, ...])	
AngularConfig(**kwargs)	
ClError	
CompositeNewtonCotesIntegrator(x, y, order)	
Cosmology([params, flat, model_name, ...])	Class for managing the cosmology.
ForecastConfig([input_xml, input_data_dir])	
H5Cl([probe1, probe2])	
H5ClCollection(**kwargs)	
Path(*args, **kwargs)	PurePath subclass that can make system calls.
PhysicalParametersCollection([...])	
RedshiftDensity([probe])	

AngularCoefficient

class seyfert.cosmology.c_ells.**AngularCoefficient**(*probe1=None*, *probe2=None*, *kernel=None*, *forecast_config=None*, *angular_config=None*, *full_output=False*)

Bases: object

Class representing a particular kind of angular power spectrum (i.e. auto-correlation or cross-correlation. These are computed in the Limber approximation, meaning that a single redshift integral is performed. Attributes ———-

Parameters

- **c_lij** – np 3D array (l, i, j) storing the values of the Cl tomographic matrix. Here l is the multipole
- **l_bin_centers** – the multipole bin centers where to compute the Cl, np 1D array.
- **l_bin_widths** – the multipole bin widths, np 1D array.

- **limber_power_spectrum_l_z** – np 2D array (l, z) storing the matter power spectrum evaluated in the Limber approximation, i.e. at wave-numbers $k = \frac{l+1/2}{r(z)}$. The redshift dimension is define by the redshift grid
- **cosmology** – instance of `Cosmology` class storing the current cosmology.
- **angular_config** (`Optional[AngularConfig]`) – configuration for the Cl computation, instance of `AngularConfig`.
- **forecast_config** (`Optional[ForecastConfig]`) – forecast configuration, instance of `ForecastConfig`.
- **kernel** (`Optional[KernelFunction]`) – kernel function to be integrated against the Limber power spectrum, instance of `KernelFunction`.

Attributes Summary

integ_method	**rtype** `str`
is_auto_correlation	**rtype** `bool`
n_ell	**rtype** `int`
n_i	**rtype** `int`
n_j	**rtype** `int`
n_z	**rtype** `int`
obs_key	**rtype** `str`
power_spectrum	Reference to the power spectrum possessed by the cosmology attribute.
weight1	Reference to first weight function making up the kernel
weight2	Reference to second weight function making up the kernel
z_array	Reference to the *z_grid* attribute of the possessed *cosmology* instance.

Methods Summary

applyVoidsCutToLimberPowerSpectrum()	**rtype** None
computeClIntegral(integrand, axis)	Method for computing a single Cl_ij.
evaluateAngularCorrelation()	Method for evaluating and storing the Cl.
evaluateLimberApproximatedPowerSpectrum()	**rtype** None
fromHDF5(file, probe1, probe2[, root])	**rtype** *AngularCoefficient*
getLimberRedshiftIntegrand(i, j)	**rtype** ndarray
getWeightFunction(probe)	**rtype** *WeightFunction*
loadCosmology(pmm_file[, load_power_spectrum])	**rtype** None
loadFromHDF5(file[, probe1, probe2, root])	**rtype** None
plot([what, axes])	
saveToHDF5(file[, root])	**rtype** None
setUp()	**rtype** None

Attributes Documentation

integ_method

Return type str

is_auto_correlation

Return type bool

n_ell

Return type int

n_i

Return type int

n_j

Return type int

n_z

Return type int

obs_key

Return type str

power_spectrum
Reference to the power spectrum possessed by the cosmology attribute.

Return type *PowerSpectrum*

weight1
Reference to first weight function making up the kernel

Return type *WeightFunction*

weight2
Reference to second weight function making up the kernel

Return type *WeightFunction*

z_array: numpy.ndarray
Reference to the *z_grid* attribute of the possessed *cosmology* instance.

Return type ndarray

Methods Documentation

applyVoidsCutToLimberPowerSpectrum()

Return type None

computeClIntegral(*integrand*, *axis*)
Method for computing a single Cl_ij.

Return type ndarray

evaluateAngularCorrelation()
Method for evaluating and storing the Cl.

Return type None

evaluateLimberApproximatedPowerSpectrum()

Return type None

classmethod fromHDF5(*file*, *probe1*, *probe2*, *root='/'*)

Return type *AngularCoefficient*

getLimberRedshiftIntegrand(*i*, *j*)

Return type ndarray

getWeightFunction(*probe*)

Return type *WeightFunction*

loadCosmology(*pmm_file*, *load_power_spectrum=True*)

Return type None

loadFromHDF5(*file*, *probe1=None*, *probe2=None*, *root='/'*)

Return type None

plot(*what='cl'*, *axes=None*, ***kwargs*)

saveToHDF5(*file*, *root='/'*)

Return type None

setUp()

Return type None

AngularCoefficientsCollector

class seyfert.cosmology.c_ells.**AngularCoefficientsCollector**(*phys_params=None*, *cosmology=None*, *fiducial_cosmology=None*, *forecast_config=None*, *angular_config=None*, *full_output=False*, *niz_file=None*)

Bases: object

Attributes Summary

probes	**rtype** List[str]
probes_combinations	**rtype** List[Tuple[str]]
short_keys	**rtype** List[str]

Methods Summary

evaluateAngularCoefficients()	**rtype** None
fromHDF5(file[, root])	**rtype** *AngularCoefficientsCollector*

continues on next page

Table 53 – continued from previous page

loadFromHDF5(file[, root])	**rtype** None
saveToHDF5(file[, root])	**rtype** None
setUp([densities])	

Attributes Documentation

probes

Return type List[str]

probes_combinations

Return type List[Tuple[str]]

short_keys

Return type List[str]

Methods Documentation

evaluateAngularCoefficients()

Return type None

classmethod fromHDF5(*file*, *root='/'*)

Return type *AngularCoefficientsCollector*

loadFromHDF5(*file*, *root='/'*)

Return type None

saveToHDF5(*file*, *root='/'*)

Return type None

setUp(*densities=None*)

ClError

exception seyfert.cosmology.c_ells.**ClError**

H5Cl

class seyfert.cosmology.c_ells.**H5Cl**(*probe1=None*, *probe2=None*, ***kwargs*)
Bases: seyfert.file_io.hdf5_io.AbstractH5FileIO

Attributes Summary

absolute_main_path	**rtype** str
main_path_rel_to_root	**rtype** str

Methods Summary

readBuildingData()	**rtype** None
writeObjectToFile(obj)	**rtype** None
writeToObject(obj)	**rtype** None

Attributes Documentation

absolute_main_path

Return type str

main_path_rel_to_root

Return type str

Methods Documentation

readBuildingData()

Return type None

writeObjectToFile(*obj*)

Return type None

writeToObject(*obj*)

Return type None

H5ClCollection

class seyfert.cosmology.c_ells.**H5ClCollection**(***kwargs*)

Bases: seyfert.file_io.hdf5_io.AbstractH5FileIO

Methods Summary

writeObjectToFile(obj)	**rtype** None
writeToObject(obj)	**rtype** None

Methods Documentation

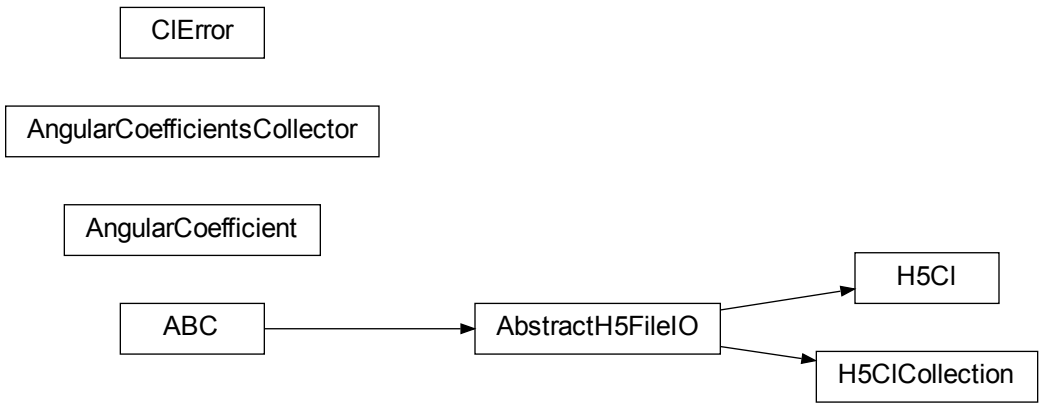
writeObjectToFile(*obj*)

Return type None

writeToObject(*obj*)

Return type None

1.8.1.3 Class Inheritance Diagram



2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z