hierArc Documentation

Release 1.1.1

Simon Birrer

Nov 22, 2022

CONTENTS:

1 Features **3**

2 Installation **5**

3 Usage **7**

4 Credits **9**
- 4.1 Installation . . . 9
- 4.2 Usage . . . 10
- 4.3 hierarc . . . 10
- 4.4 Contributing . . . 43
- 4.5 Credits . . . 46
- 4.6 History . . . 46
- 4.7 Indices and tables . . . 46

Python Module Index **47**

Index **49**

Hierarchical analysis of strong lensing systems to infer lens properties and cosmological parameters simultaneously.

The software is originated from Birrer et al. 2020 and is in active development.

- Free software: BSD license
- Documentation: https://hierarc.readthedocs.io.

CHAPTER

ONE

FEATURES

The software allows to fit lenses with measured time delays, imaging information, kinematics constraints and standardizable magnifications with parameters described on the ensemble level.

CHAPTER

TWO

INSTALLATION

```
$ pip install hierarc --user
```

CHAPTER

THREE

USAGE

The full analysis of Birrer et al. 2020 is publicly available at this TDCOSMO repository . A forecast based on hierArc is presented by Birrer & Treu 2020 and the notebooks are available at this repository. The extension to using hierArc with standardizable magnifications is presented by Birrer et al. 2021 and the forecast analysis is publicly available here. For example use cases we refer to the notebooks of these analyses.

CHAPTER

FOUR

CREDITS

Simon Birrer & the TDCOSMO team.

Please cite Birrer et al. 2020 if you make use of this software for your research.

4.1 Installation

4.1.1 Stable release

To install hierArc, run this command in your terminal:

```
$ pip install hierarc
```

This is the preferred method to install hierArc, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

4.1.2 From sources

The sources for hierArc can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/sibirrer/hierarc
```

Or download the tarball:

```
$ curl -OJL https://github.com/sibirrer/hierarc/tarball/main
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

4.2 Usage

To use hierArc in a project:

```
import hierarc
```

4.3 hierarc

4.3.1 hierarc package

Subpackages

hierarc.Diagnostics package

Submodules

hierarc.Diagnostics.goodness_of_fit module

class hierarc.Diagnostics.goodness_of_fit.**GoodnessOfFit**(*kwargs_likelihood_list*)

Bases: object

class to manage goodness of fit diagnostics

kin_fit(*cosmo*, *kwargs_lens*, *kwargs_kin*)

plots the prediction and the uncorrelated error bars on the individual lenses currently works for likelihood classes 'TDKinGaussian', 'KinGaussian'

Parameters

- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – lens model parameter keyword arguments
- **kwargs_kin** – kinematics model keyword arguments

Returns

list of name, measurement, measurement errors, model prediction, model prediction error

plot_ddt_fit(*cosmo*, *kwargs_lens*, *kwargs_kin*, *color_measurement=None*, *color_prediction=None*, *redshift_trend=False*)

plots the prediction and the uncorrelated error bars on the individual lenses currently works for likelihood classes 'TDKinGaussian', 'KinGaussian'

Parameters

- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – lens model parameter keyword arguments
- **kwargs_kin** – kinematics model keyword arguments
- **color_measurement** – color of measurement
- **color_prediction** – color of model prediction
- **redshift_trend** – boolean, if True, plots as a function of redshift

Returns
fig, axes of matplotlib instance

`plot_ifu_fit`(*ax*, *cosmo*, *kwargs_lens*, *kwargs_kin*, *lens_index*, *bin_edges*, *show_legend=True*, *color_measurement=None*, *color_prediction=None*)

plot an individual IFU data goodness of fit

Parameters

- **ax** – matplotlib axes instance
- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – lens model parameter keyword arguments
- **kwargs_kin** – kinematics model keyword arguments
- **lens_index** – int, index in kwargs_lens to be plotted (needs to be of type 'IFUKinCov')
- **bin_edges** (*numpy array or float.*) – radial bin edges in arc seconds. If number, then uniform bin_edges sampled from 0
- **show_legend** – bool, to show legend
- **color_measurement** – color of measurement
- **color_prediction** – color of model prediction

Returns
figure as axes instance

`plot_kin_fit`(*cosmo*, *kwargs_lens*, *kwargs_kin*, *color_measurement=None*, *color_prediction=None*)

plots the prediction and the uncorrelated error bars on the individual lenses currently works for likelihood classes 'TDKinGaussian', 'KinGaussian'

Parameters

- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – lens model parameter keyword arguments
- **kwargs_kin** – kinematics model keyword arguments
- **color_measurement** – color of measurement
- **color_prediction** – color of model prediction

Returns
fig, axes of matplotlib instance

`reduced_chi2`(*cosmo*, *kwargs_lens*, *kwargs_kin*)

reduced chi^2 of fit

Module contents

hierarc.LensPosterior package

Submodules

hierarc.LensPosterior.anisotropy_config module

class hierarc.LensPosterior.anisotropy_config.**AnisotropyConfig**(*anisotropy_model*, *r_eff*)

Bases: object

class to manage the anisotropy model and parameters for the Posterior processing

property **ani_param_array**

Returns
numpy array of anisotropy parameter values to be explored

anisotropy_kwargs(*a_ani*, *beta_inf=None*)

Parameters

- **a_ani** – anisotropy parameter
- **beta_inf** – anisotropy at infinity (only used for 'GOM' model)

Returns
list of anisotropy keyword arguments, value of anisotropy parameter list

property **kwargs_anisotropy_base**

Returns
keyword arguments of base anisotropy model configuration

hierarc.LensPosterior.base_config module

class hierarc.LensPosterior.base_config.**BaseLensConfig**(*z_lens*, *z_source*, *theta_E*, *theta_E_error*, *gamma*, *gamma_error*, *r_eff*, *r_eff_error*, *kwargs_aperture*, *kwargs_seeing*, *kwargs_numerics_galkin*, *anisotropy_model*, *kwargs_lens_light=None*, *lens_light_model_list=['HERNQUIST']*, *MGE_light=False*, *kwargs_mge_light=None*, *hernquist_approx=True*, *sampling_number=1000*, *num_psf_sampling=100*, *num_kin_sampling=1000*, *multi_observations=False*)

Bases: TDCosmography, *ImageModelPosterior*, *AnisotropyConfig*

this class contains and manages the base configurations of the lens posteriors and makes sure that they are universally applied consistently through the different likelihood definitions

hierarc.LensPosterior.ddt_kin_constraints module

class hierarc.LensPosterior.ddt_kin_constraints.**DdtKinConstraints**(*z_lens*, *z_source*, *ddt_samples*, *ddt_weights*, *theta_E*, *theta_E_error*, *gamma*, *gamma_error*, *r_eff*, *r_eff_error*, *sigma_v_measured*, *kwargs_aperture*, *kwargs_seeing*, *kwargs_numerics_galkin*, *anisotropy_model*, *sigma_v_error_independent=None*, *sigma_v_error_covariant=None*, *sigma_v_error_cov_matrix=None*, *kwargs_lens_light=None*, *lens_light_model_list=['HERNQUIST']*, *MGE_light=False*, *kwargs_mge_light=None*, *hernquist_approx=True*, *kappa_ext=0*, *kappa_ext_sigma=0*, *sampling_number=1000*, *num_psf_sampling=100*, *num_kin_sampling=1000*, *multi_observations=False*)

Bases: *KinConstraints*

class for sampling Ds/Dds posteriors from imaging data and kinematic constraints with additional constraints on the time-delay distance Ddt

hierarchy_configuration(*num_sample_model=20*)

routine to configure the likelihood to be used in the hierarchical sampling. In particular, a default configuration is set to compute the Gaussian approximation of Ds/Dds by sampling the posterior and the estimate of the variance of the sample. The anisotropy scaling is then performed. Different anisotropy models are supported.

Parameters

num_sample_model – number of samples drawn from the lens and light model posterior to compute the dimensionless kinematic component J()

Returns

keyword arguments

hierarc.LensPosterior.ddt_kin_gauss_constraints module

class hierarc.LensPosterior.ddt_kin_gauss_constraints.**DdtGaussKinConstraints**(*z_lens*, *z_source*, *ddt_mean*, *ddt_sigma*, *theta_E*, *theta_E_error*, *gamma*, *gamma_error*, *r_eff*, *r_eff_error*, *sigma_v_measured*, *kwargs_aperture*, *kwargs_seeing*, *kwargs_numerics_galkin*, *anisotropy_model*, *sigma_v_error_independent=None*, *sigma_v_error_covariant=None*, *sigma_v_error_cov_matrix=None*, *kwargs_lens_light=None*, *lens_light_model_list=['HERNQU*, *MGE_light=False*, *kwargs_mge_light=None*, *hernquist_approx=True*, *kappa_ext=0*, *kappa_ext_sigma=0*, *sampling_number=1000*, *num_psf_sampling=100*, *num_kin_sampling=1000*, *multi_observations=False*)

Bases: *KinConstraints*

class for sampling Ds/Dds posteriors from imaging data and kinematic constraints with additional constraints on the time-delay distance Ddt

hierarchy_configuration(*num_sample_model=20*)

routine to configure the likelihood to be used in the hierarchical sampling. In particular, a default configuration is set to compute the Gaussian approximation of Ds/Dds by sampling the posterior and the estimate of the variance of the sample. The anisotropy scaling is then performed. Different anisotropy models are supported.

Parameters

num_sample_model – number of samples drawn from the lens and light model posterior to compute the dimensionless kinematic component J()

Returns

keyword arguments

hierarc.LensPosterior.imaging_constraints module

class hierarc.LensPosterior.imaging_constraints.**ImageModelPosterior**(*theta_E*, *theta_E_error*, *gamma*, *gamma_error*, *r_eff*, *r_eff_error*)

Bases: object

class to manage lens and light model posteriors inferred from imaging data

draw_lens(*no_error=False*)

Parameters
no_error – bool, if True, does not render from the uncertainty but uses the mean values instead

Returns
theta_E, gamma, r_eff, delta_r_eff

hierarc.LensPosterior.kin_constraints module

class hierarc.LensPosterior.kin_constraints.**KinConstraints**(*z_lens*, *z_source*, *theta_E*, *theta_E_error*, *gamma*, *gamma_error*, *r_eff*, *r_eff_error*, *sigma_v_measured*, *kwargs_aperture*, *kwargs_seeing*, *kwargs_numerics_galkin*, *anisotropy_model*, *sigma_v_error_independent=None*, *sigma_v_error_covariant=None*, *sigma_v_error_cov_matrix=None*, *kwargs_lens_light=None*, *lens_light_model_list=['HERNQUIST']*, *MGE_light=False*, *kwargs_mge_light=None*, *hernquist_approx=True*, *sampling_number=1000*, *num_psf_sampling=100*, *num_kin_sampling=1000*, *multi_observations=False*)

Bases: *BaseLensConfig*

class that manages constraints from Integral Field Unit spectral observations.

anisotropy_scaling()

Returns
anisotropy scaling grid along the axes defined by ani_param_array

property error_cov_measurement

error covariance matrix of the measured velocity dispersion data points This is either calculated from the diagonal 'sigma_v_error_independent' and the off-diagonal 'sigma_v_error_covariant' terms, or directly from the 'sigma_v_error_cov_matrix' if provided.

Returns
nxn matrix of the error covariances in the velocity dispersion measurements (km/s)^2

hierarchy_configuration(*num_sample_model=20*)

routine to configure the likelihood to be used in the hierarchical sampling. In particular, a default configuration is set to compute the Gaussian approximation of Ds/Dds by sampling the posterior and the estimate of the variance of the sample. The anisotropy scaling is then performed. Different anisotropy models are supported.

Parameters
num_sample_model – number of samples drawn from the lens and light model posterior to compute the dimensionless kinematic component J()

Returns
keyword arguments

j_kin_draw(*kwargs_anisotropy*, *no_error=False*)

one simple sampling realization of the dimensionless kinematics of the model

Parameters

- **kwargs_anisotropy** – keyword argument of anisotropy setting
- **no_error** – bool, if True, does not render from the uncertainty but uses the mean values instead

Returns
dimensionless kinematic component J() Birrer et al. 2016, 2019

model_marginalization(*num_sample_model=20*)

Parameters
num_sample_model – number of samples drawn from the lens and light model posterior to compute the dimensionless kinematic component J()

Returns
J() as array for each measurement prediction, covariance matrix in sqrt(J)

Module contents

hierarc.Likelihood package

Subpackages

hierarc.Likelihood.LensLikelihood package

Submodules

hierarc.Likelihood.LensLikelihood.base_lens_likelihood module

class hierarc.Likelihood.LensLikelihood.base_lens_likelihood.**LensLikelihoodBase**(*z_lens*, *z_source*, *likelihood_type*, *name='name'*, *normalized=False*, ***kwargs_likelihood*)

Bases: object

master class containing the likelihood definitions of different analysis

ddt_measurement()

Inferred Ddt from a lens model (i.e. power-law fit) and time-delay, without lambda correction (excludes also the external convergence contribution)

Returns
ddt measurement median, 1-sigma (without lambda correction factor)

log_likelihood(*ddt*, *dd*, *aniso_scaling=None*, *sigma_v_sys_error=None*, *mu_intrinsic=None*)

Parameters

- **ddt** – time-delay distance [physical Mpc]
- **dd** – angular diameter distance to the lens [physical Mpc]
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.
- **sigma_v_sys_error** – unaccounted uncertainty in the velocity dispersion measurement
- **mu_intrinsic** – float, intrinsic source brightness (in magnitude)

Returns
natural logarithm of the likelihood of the data given the model

num_data()

number of data points across the lens sample

Returns
integer

sigma_v_measurement(*sigma_v_sys_error=None*)

Returns
data vector, measurement covariance matrix for velocity dispersion

sigma_v_prediction(*ddt*, *dd*, *aniso_scaling=None*)

Parameters

- **ddt** – ddt in physical Mpc
- **dd** – dd in physical Mpc
- **aniso_scaling** – anisotropy scaling in J

Returns
model predicted velocity dispersion (vector) and model covariance matrix thereof

hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood module

class hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood.**DdtDdGaussian**(*z_lens*, *z_source*, *ddt_mean*, *ddt_sigma*, *dd_mean*, *dd_sigma*)

Bases: object

class for joint kinematics and time delay likelihood assuming independent Gaussian likelihoods in Ddt and Dd. Attention: Gaussian errors in the velocity dispersion do not translate into Gaussian uncertainties in Dd.

ddt_measurement()

Returns
mean, 1-sigma of the ddt inference/model measurement

log_likelihood(*ddt*, *dd*, *aniso_scaling=None*)

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.

Returns
log likelihood given the single lens analysis

hierarc.Likelihood.LensLikelihood.ddt_dd_kde_likelihood module

class hierarc.Likelihood.LensLikelihood.ddt_dd_kde_likelihood.**DdtDdKDELikelihood**(*z_lens*, *z_source*, *dd_samples*, *ddt_samples*, *kde_type='scipy_gaussian'*, *bandwidth=1*, *interpol=False*, *num_interp_grid=100*)

Bases: object

class for evaluating the 2-d posterior of Ddt vs Dd coming from a lens with time delays and kinematics measurement

log_likelihood(*ddt*, *dd*, *aniso_scaling=None*)

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector

- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.

Returns
log likelihood given the single lens analysis

hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood module

class hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood.**DdtGaussKinLikelihood**(*z_lens*, *z_source*, *ddt_mean*, *ddt_sigma*, *sigma_v_measuremen*, *j_model*, *error_cov_measuremen*, *error_cov_j_sqrt*, *sigma_sys_error_inc*, *normalized=True*)

Bases: object

class for joint kinematics and time delay likelihood assuming that they are independent Uses KinLikelihood and DdtHistLikelihood combined

ddt_measurement()

Returns
mean, 1-sigma of the ddt inference/model measurement

log_likelihood(*ddt*, *dd*, *aniso_scaling=None*, *sigma_v_sys_error=None*, *sigma_v_sys_offset=None*)

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.
- **sigma_v_sys_error** – float (optional) added error on the velocity dispersion measurement in quadrature
- **sigma_v_sys_offset** – float (optional) for a fractional systematic offset in the kinematic measurement such that sigma_v = sigma_v_measured * (1 + sigma_v_sys_offset)

Returns
log likelihood given the single lens analysis

sigma_v_measurement(*sigma_v_sys_error=None*, *sigma_v_sys_offset=None*)

Parameters

- **sigma_v_sys_error** – float (optional) added error on the velocity dispersion measurement in quadrature
- **sigma_v_sys_offset** – float (optional) for a fractional systematic offset in the kinematic measurement such that sigma_v = sigma_v_measured * (1 + sigma_v_sys_offset)

Returns
measurement mean (vector), measurement covariance matrix

sigma_v_prediction(*ddt*, *dd*, *aniso_scaling=1*)

model prediction mean velocity dispersion vector and model prediction covariance matrix

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.

Returns
model prediction mean velocity dispersion vector and model prediction covariance matrix

hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood module

class hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood.**DdtGaussianLikelihood**(*z_lens*, *z_source*, *ddt_mean*, *ddt_sigma*)

Bases: object

class to handle cosmographic likelihood coming from modeling lenses with imaging and kinematic data but no time delays. Thus Ddt is not constrained but the kinematics can constrain Ds/Dds

The current version includes a Gaussian in Ds/Dds but can be extended.

ddt_measurement()

Returns
mean, 1-sigma of the ddt inference/model measurement

log_likelihood(*ddt*, *dd=None*)

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector

Returns
log likelihood given the single lens analysis

hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood module

class hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood.**DdtHistKinLikelihood**(*z_lens*, *z_source*, *ddt_samples*, *sigma_v_measurement*, *j_model*, *error_cov_measurement*, *error_cov_j_sqrt*, *ddt_weights=None*, *kde_kernel='gaussian'*, *bandwidth=20*, *nbins_hist=200*, *sigma_sys_error_include_normalized=True*)

Bases: `object`

class for joint kinematics and time delay likelihood assuming that they are independent Uses KinLikelihood and DdtHistLikelihood combined

ddt_measurement()

Returns
mean, 1-sigma of the ddt inference/model measurement

log_likelihood(*ddt*, *dd*, *aniso_scaling=None*, *sigma_v_sys_error=None*)

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – numpy array of anisotropy scaling on prediction of Ds/Dds

Returns
log likelihood given the single lens analysis

sigma_v_measurement(*sigma_v_sys_error=None*, *sigma_v_sys_offset=None*)

Parameters

- **sigma_v_sys_error** – float (optional) added error on the velocity dispersion measurement in quadrature
- **sigma_v_sys_offset** – float (optional) for a fractional systematic offset in the kinematic measurement such that sigma_v = sigma_v_measured * (1 + sigma_v_sys_offset)

Returns
measurement mean (vector), measurement covariance matrix

sigma_v_prediction(*ddt*, *dd*, *aniso_scaling=1*)

model prediction mean velocity dispersion vector and model prediction covariance matrix

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.

Returns
model prediction mean velocity dispersion vector and model prediction covariance matrix

hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood module

class hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood.**DdtHistKDELikelihood**(*z_lens*, *z_source*, *ddt_samples*, *kde_kernel='gaussian'*, *ddt_weights=None*, *bandwidth=20*, *nbins_hist=200*, *normalized=False*)

Bases: object

Evaluates the likelihood of a time-delay distance ddt (in Mpc) against the model predictions, using a loglikelihood sampled from a Kernel Density Estimator. the KDE is constructed using a binned version of the full samples. Greatly improves speed at the cost of a (tiny) loss in precision

__warning:: you should adjust bandwidth and nbins_hist to the spacing and size of your samples chain!

original source: https://github.com/shsuyu/H0LiCOW-public/blob/master/H0_inference_code/lensutils.py credits to Martin Millon, Aymeric Galan

ddt_measurement()

Returns
mean, 1-sigma of the ddt inference/model measurement

log_likelihood(*ddt*, *dd=None*)

Note: kinematics + imaging data can constrain Ds/Dds. The input of Ddt, Dd is transformed here to match Ds/Dds

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector

Returns
log likelihood given the single lens analysis

class hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood.**DdtHistLikelihood**(*z_lens*, *z_source*, *ddt_samples*, *ddt_weights=None*, *nbins_hist=200*, *normalized=False*, *binning_method=None*)

Bases: object

Evaluates the likelihood of a time-delay distance ddt (in Mpc) against the model predictions, using a loglikelihood sampled from a Kernel Density Estimator. The KDE is constructed using a binned version of the full samples. Greatly improves speed at the cost of a (tiny) loss in precision.

Warning: you should adjust bandwidth and nbins_hist to the spacing and size of your samples chain!

original source: https://github.com/shsuyu/H0LiCOW-public/blob/master/H0_inference_code/lensutils.py credits to Martin Millon, Aymeric Galan

ddt_measurement()

Returns
mean, 1-sigma of the ddt inference/model measurement

log_likelihood(*ddt*, *dd=None*)

Note: kinematics + imaging data can constrain Ds/Dds. The input of Ddt, Dd is transformed here to match Ds/Dds

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector

Returns
log likelihood given the single lens analysis

hierarc.Likelihood.LensLikelihood.ddt_lognorm_likelihood module

class hierarc.Likelihood.LensLikelihood.ddt_lognorm_likelihood.**DdtLogNormLikelihood**(*z_lens*, *z_source*, *ddt_mu*, *ddt_sigma*)

Bases: object

The cosmographic likelihood coming from modeling lenses with imaging and kinematic data but no time delays, where the form of the likelihood is a lognormal distribution. Thus Ddt is not constrained but the kinematics can constrain Ds/Dds

The current version includes a Gaussian in Ds/Dds but can be extended.

log_likelihood(*ddt*, *dd=None*)

Note: kinematics + imaging data can constrain Ds/Dds. The input of Ddt, Dd is transformed here to match Ds/Dds

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector

Returns
log likelihood given the single lens analysis

hierarc.Likelihood.LensLikelihood.ds_dds_gauss_likelihood module

class hierarc.Likelihood.LensLikelihood.ds_dds_gauss_likelihood.**DsDdsGaussianLikelihood**(*z_lens*, *z_source*, *ds_dds_mean*, *ds_dds_sigma*)

Bases: object

class to handle cosmographic likelihood coming from modeling lenses with imaging and kinematic data but no time delays. Thus Ddt is not constrained but the kinematics can constrain Ds/Dds. The likelihood in Ds/Dds is assumed Gaussian. Attention: Gaussian uncertainties in velocity dispersion do not translate into Gaussian uncertainties in Ds/Dds.

log_likelihood(*ddt*, *dd*, *aniso_scaling=None*)

Note: kinematics + imaging data can constrain Ds/Dds. The input of Ddt, Dd is transformed here to match Ds/Dds

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.

Returns
log likelihood given the single lens analysis

hierarc.Likelihood.LensLikelihood.kin_likelihood module

class hierarc.Likelihood.LensLikelihood.kin_likelihood.**KinLikelihood**(*z_lens*, *z_source*, *sigma_v_measurement*, *j_model*, *error_cov_measurement*, *error_cov_j_sqrt*, *normalized=True*, *sigma_sys_error_include=False*)

Bases: object

likelihood to deal with IFU kinematics constraints with covariances in both the model and measured velocity dispersion

cov_error_measurement(*sigma_v_sys_error=None*)

Parameters
sigma_v_sys_error – float (optional) added error on the velocity dispersion measurement in quadrature

Returns
error covariance matrix of the velocity dispersion measurements

cov_error_model(*ds_dds*, *aniso_scaling=1*)

Parameters

- **ds_dds** – Ds/Dds
- **aniso_scaling** – scaling of the anisotropy affecting sigma_v^2

Returns
covariance matrix of the error in the predicted model (from mass model uncertainties)

log_likelihood(*ddt*, *dd*, *aniso_scaling=None*, *sigma_v_sys_error=None*, *sigma_v_sys_offset=None*)

Note: kinematics + imaging data can constrain Ds/Dds. The input of Ddt, Dd is transformed here to match Ds/Dds

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.
- **sigma_v_sys_error** – float (optional) added error on the velocity dispersion measurement in quadrature
- **sigma_v_sys_offset** – float (optional) for a fractional systematic offset in the kinematic measurement such that sigma_v = sigma_v_measured * (1 + sigma_v_sys_offset)

Returns
log likelihood given the single lens analysis

sigma_v_measurement(*sigma_v_sys_error=None*, *sigma_v_sys_offset=None*)

Parameters

- **sigma_v_sys_error** – float (optional) added error on the velocity dispersion measurement in quadrature
- **sigma_v_sys_offset** – float (optional) for a fractional systematic offset in the kinematic measurement such that sigma_v = sigma_v_measured * (1 + sigma_v_sys_offset)

Returns
measurement mean (vector), measurement covariance matrix

sigma_v_measurement_mean(*sigma_v_sys_offset=None*)

Parameters
sigma_v_sys_offset – float (optional) for a fractional systematic offset in the kinematic measurement such that sigma_v = sigma_v_measured * (1 + sigma_v_sys_offset)

Returns
corrected measured velocity dispersion

sigma_v_model(*ds_dds*, *aniso_scaling=1*)

model predicted velocity dispersion for the IFU's

Parameters

- **ds_dds** – Ds/Dds
- **aniso_scaling** – scaling of the anisotropy affecting sigma_v^2

Returns
array of predicted velocity dispersions

sigma_v_prediction(*ddt*, *dd*, *aniso_scaling=1*)

model prediction mean velocity dispersion vector and model prediction covariance matrix

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **aniso_scaling** – array of size of the velocity dispersion measurement or None, scaling of the predicted dimensionless quantity J (proportional to sigma_v^2) of the anisotropy model in the sampling relative to the anisotropy model used to derive the prediction and covariance matrix in the init of this class.

Returns
model prediction mean velocity dispersion vector and model prediction covariance matrix

hierarc.Likelihood.LensLikelihood.mag_likelihood module

class hierarc.Likelihood.LensLikelihood.mag_likelihood.**MagnificationLikelihood**(*amp_measured*, *cov_amp_measured*, *magnification_model*, *cov_magnification_model*, *magnitude_zero_point=20*)

Bases: object

likelihood of an unlensed apprarent source magnification given a measurement of the magnified brightness This can i.e. be applied to lensed SNIa on the population level

log_likelihood(*mu_intrinsic*)

Parameters
mu_intrinsic – intrinsic brightness of the source (already incorporating the inverse MST transform)

Returns
log likelihood of the measured magnified images given the source brightness

hierarc.Likelihood.LensLikelihood.td_mag_likelihood module

class hierarc.Likelihood.LensLikelihood.td_mag_likelihood.**TDMagLikelihood**(*time_delay_measured*, *cov_td_measured*, *amp_measured*, *cov_amp_measured*, *fermat_diff*, *magnification_model*, *cov_model*, *magnitude_zero_point=20*)

Bases: object

likelihood of time delays and magnification likelihood

This likelihood uses linear flux units and linear lensing magnifications.

log_likelihood(*ddt*, *mu_intrinsic*)

Parameters

- **ddt** – time-delay distance (physical Mpc)
- **mu_intrinsic** – intrinsic brightness of the source (already incorporating the inverse MST transform)

Returns
log likelihood of the measured magnified images given the source brightness

hierarc.Likelihood.LensLikelihood.td_mag_magnitude_likelihood module

class hierarc.Likelihood.LensLikelihood.td_mag_magnitude_likelihood.**TDMagMagnitudeLikelihood**(*time_delay_m*
cov_td_measu
magnitude_measure
cov_magnitud
fermat_diff, *magnification_model*, *cov_model*)

Bases: object

likelihood of time delays and magnification likelihood

This likelihood uses astronomical magnitude units in flux measurement and lensing magnification and Gaussian uncertainties in this space.

log_likelihood(*ddt*, *mu_intrinsic*)

Parameters

- **ddt** – time-delay distance (physical Mpc)

- **mu_intrinsic** – intrinsic brightness of the source (already incorporating the inverse MST transform)

Returns
log likelihood of the measured magnified images given the source brightness

Module contents

hierarc.Likelihood.SneLikelihood package

Submodules

hierarc.Likelihood.SneLikelihood.sne_likelihood module

class hierarc.Likelihood.SneLikelihood.sne_likelihood.**SneLikelihood**(*sample_name='CUSTOM'*, ***kwargs_sne_likelihood*)

Bases: object

Supernovae likelihood This class supports custom likelihoods as well as likelihoods from the Pantheon sample from file

log_likelihood(*cosmo*, *apparent_m_z=None*, *sigma_m_z=None*, *z_anchor=0.1*)

Parameters

- **cosmo** – instance of a class to compute angular diameter distances on arrays
- **apparent_m_z** – mean apparent magnitude of SN Ia at z=z_anchor (optional)
- **z_anchor** – redshift where definition of apparent_m_z is set (only applicable when apparent_m_z != None)
- **sigma_m_z** – 1-sigma scatter in magnitude in the intrinsic SNe brightness distribution not accounted-for by the covariance matrix

Returns
log likelihood of the data given the specified cosmology

hierarc.Likelihood.SneLikelihood.sne_likelihood_custom module

class hierarc.Likelihood.SneLikelihood.sne_likelihood_custom.**CustomSneLikelihood**(*mag_mean*, *cov_mag*, *zhel*, *zcmb*, *no_intrinsic_scatter=False*)

Bases: object

class method for an arbitrary apparent magnitude likelihood of a Sne sample where the error and systematic covariance matrix is described in astronomical magnitude space

log_likelihood_lum_dist(*lum_dists*, *estimated_scriptm=None*, *sigma_m_z=None*)

Parameters

- **lum_dists** – numpy array of luminosity distances to the measured supernovae bins (units do not matter since normalization is subtracted off for the likelihood)
- **estimated_scriptm** – mean magnitude at lum_dist=0 (optional)

- **sigma_m_z** – 1-sigma scatter in magnitude in the intrinsic SNe brightness distribution not accounted-for by the covariance matrix

Returns
log likelihood of the data given the luminosity distances

hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file module

This is a lightweight version of the COSMOMC/Cobaya sampler: https://github.com/CobayaSampler/cobaya/blob/71b87842d12c6a04eec182c39b6bef1cd9a987af/cobaya/likelihoods/_base_classes/_sn_prototype.py#L287 It uses the binned Pantheon data: https://github.com/dscolnic/Pantheon/blob/master/Binned_data/lcparam_DS17f.txt And computes the cosmographic likelihood. The main difference is that this class is compatible with the hierArc cosmology module for evaluating likelihoods. This likelihood does NOT include systematics!

- If you use `sn.pantheon`, please cite: Scolnic, D. M. et al, 2018 *The Complete Light-curve Sample of Spectroscopically Confirmed Type Ia Supernovae from Pan-STARRS1 and Cosmological Constraints from The Combined Pantheon Sample* (arXiv:1710.00845)

Synopsis
Supernovae likelihood, from CosmoMC's JLA module, for Pantheon and JLA samples.

Author
Alex Conley, Marc Betoule, Antony Lewis (see source for more specific authorship)

`class hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file.SneLikelihoodFromFile(sample_name='Panthe pec_z=0.001)`

Bases: `object`

Base likelihood class for evaluating Sne likelihoods

`log_likelihood_lum_dist(lum_dists, estimated_scriptm=None, sigma_m_z=None)`

Parameters

- **lum_dists** – numpy array of luminosity distances to the measured supernovae bins (units do not matter since normalization is subtracted off for the likelihood)
- **estimated_scriptm** – mean magnitude at lum_dist=0 (optional)
- **sigma_m_z** – 1-sigma scatter in magnitude in the intrinsic SNe brightness distribution not accounted-for by the covariance matrix. This variable is not supported in the current implementation of the Pantheon sample

Returns
log likelihood of the data given the luminosity distances

`hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file.read_covariance_matrix(filename, nsn)`

reads in covariance matrix file and returns it as a numpy matrix

Parameters

- **filename** – string, absolute path of covariance matrix file
- **nsn** – number of supernovae (or bins)

Returns
nxn covariance matrix

hierarc.Likelihood.SneLikelihood.sne_pantheon_plus module

class hierarc.Likelihood.SneLikelihood.sne_pantheon_plus.**PantheonPlusData**

Bases: object

This class is a lightweight version of the Pantheon+ analysis presented in Pantheon+ likelihood.

The data covariances that are stored in hierArc are originally from Pantheon+ Data products.

If you make use of these products, please cite Brout et al. 2022

build_covariance()

Run once at the start to build the covariance matrix for the data

Module contents

Submodules

hierarc.Likelihood.anisotropy_scaling module

class hierarc.Likelihood.anisotropy_scaling.**AnisotropyScalingIFU**(*anisotropy_model='NONE'*, *ani_param_array=None*, *ani_scaling_array_list=None*)

Bases: object

class to manage anisotropy scalings for IFU data

ani_scaling(*aniso_param_array*)

Parameters
aniso_param_array – anisotropy parameter array

Returns
scaling J(a_ani) for the IFU's

draw_anisotropy(*a_ani=None*, *a_ani_sigma=0*, *beta_inf=None*, *beta_inf_sigma=0*)

draw Gaussian distribution and re-sample if outside bounds

Parameters

- **a_ani** – mean of the distribution
- **a_ani_sigma** – std of the distribution
- **beta_inf** – anisotropy at infinity (relevant for GOM model)
- **beta_inf_sigma** – std of beta_inf distribution

Returns
random draw from the distribution

class hierarc.Likelihood.anisotropy_scaling.**AnisotropyScalingSingleAperture**(*ani_param_array*, *ani_scaling_array*)

Bases: object

class to manage anisotropy scaling for single slit observation

ani_scaling(*aniso_param_array*)

Parameters
aniso_param_array – anisotropy parameter array

Returns
scaling J(a_ani) for single slit

hierarc.Likelihood.cosmo_likelihood module

class hierarc.Likelihood.cosmo_likelihood.**CosmoLikelihood**(*kwargs_likelihood_list*, *cosmology*, *kwargs_bounds*, *sne_likelihood=None*, *kwargs_sne_likelihood=None*, *KDE_likelihood_chain=None*, *kwargs_kde_likelihood=None*, *ppn_sampling=False*, *lambda_mst_sampling=False*, *lambda_mst_distribution='delta'*, *anisotropy_sampling=False*, *kappa_ext_sampling=False*, *kappa_ext_distribution='NONE'*, *alpha_lambda_sampling=False*, *lambda_ifu_sampling=False*, *lambda_ifu_distribution='NONE'*, *sigma_v_systematics=False*, *sne_apparent_m_sampling=False*, *sne_distribution='GAUSSIAN'*, *z_apparent_m_anchor=0.1*, *log_scatter=False*, *anisotropy_model='OM'*, *anisotropy_distribution='NONE'*, *custom_prior=None*, *interpolate_cosmo=True*, *num_redshift_interp=100*, *cosmo_fixed=None*)

Bases: object

this class contains the likelihood function of the Strong lensing analysis

cosmo_instance(*kwargs_cosmo*)

Parameters
kwargs_cosmo – cosmology parameter keyword argument list

Returns
astropy.cosmology (or equivalent interpolation scheme class)

likelihood(*args*)

Parameters
args – list of sampled parameters

Returns
log likelihood of the combined lenses

hierarc.Likelihood.hierarchy_likelihood module

class hierarc.Likelihood.hierarchy_likelihood.**LensLikelihood**(*z_lens*, *z_source*, *name='name'*, *likelihood_type='TDKin'*, *anisotropy_model='NONE'*, *ani_param_array=None*, *ani_scaling_array_list=None*, *ani_scaling_array=None*, *num_distribution_draws=50*, *kappa_ext_bias=False*, *kappa_pdf=None*, *kappa_bin_edges=None*, *mst_ifu=False*, *lambda_scaling_property=0*, *normalized=False*, ***kwargs_likelihood*)

Bases: *TransformedCosmography*, *LensLikelihoodBase*, *AnisotropyScalingIFU*

master class containing the likelihood definitions of different analysis

angular_diameter_distances(*cosmo*)

time-delay distance Ddt, angular diameter distance to the lens (dd)

Parameters

cosmo – astropy.cosmology instance (or equivalent with interpolation)

Returns

ddt, dd, ds in units physical Mpc

check_dist(*kwargs_lens*, *kwargs_kin*, *kwargs_source*)

checks if the provided keyword arguments describe a distribution function of hyper parameters or are single values

Parameters

- **kwargs_lens** – lens model hyper parameter keywords
- **kwargs_kin** – kinematic model hyper parameter keywords
- **kwargs_source** – source brightness hyper parameter keywords

Returns

bool, True if delta function, else False

ddt_dd_model_prediction(*cosmo*, *kwargs_lens=None*)

predicts the model uncertainty corrected ddt prediction of the applied model (e.g. power-law)

Parameters

- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – keywords of the hyper parameters of the lens model

Returns

ddt_model mean, ddt_model sigma, dd_model mean, dd_model sigma

draw_lens(*lambda_mst=1*, *lambda_mst_sigma=0*, *kappa_ext=0*, *kappa_ext_sigma=0*, *gamma_ppn=1*, *lambda_ifu=1*, *lambda_ifu_sigma=0*, *alpha_lambda=0*)

draws a realization of a specific model from the hyper-parameter distribution

Parameters

- **lambda_mst** – MST transform
- **lambda_mst_sigma** – spread in the distribution
- **kappa_ext** – external convergence mean in distribution
- **kappa_ext_sigma** – spread in the distribution
- **gamma_ppn** – Post-Newtonian parameter
- **lambda_ifu** – secondary lambda_mst parameter for subset of lenses specified for
- **lambda_ifu_sigma** – secondary lambda_mst_sigma parameter for subset of lenses specified for
- **alpha_lambda** – float, linear slope of the lambda_int scaling relation with lens quantity self._lambda_scaling_property

Returns
draw from the distributions

static draw_source(*mu_sne=1*, *sigma_sne=0*, *lum_dist=0*, ***kwargs*)

draws a source magnitude from a distribution specified by population parameters

Parameters

- **mu_sne** – mean magnitude of SNe
- **sigma_sne** – std of magnitude distribution of SNe relative to the mean magnitude
- **lum_dist** – luminosity distance (astronomical magnitude scaling of defined brightness to the source redshift)

Returns
realization of source amplitude given distribution

hyper_param_likelihood(*ddt*, *dd*, *delta_lum_dist*, *kwargs_lens=None*, *kwargs_kin=None*, *kwargs_source=None*)

log likelihood of the data of a lens given a model (defined with hyper-parameters) and cosmological distances

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **delta_lum_dist** – relative luminosity distance to pivot redshift
- **kwargs_lens** – keywords of the hyper parameters of the lens model
- **kwargs_kin** – keyword arguments of the kinematic model hyper parameters
- **kwargs_source** – keyword argument of the source model (such as SNe)

Returns
log likelihood given the single lens analysis for the given hyper parameter

lens_log_likelihood(*cosmo*, *kwargs_lens=None*, *kwargs_kin=None*, *kwargs_source=None*)

log likelihood of the data of a lens given a model (defined with hyper-parameters) and cosmology

Parameters

- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – keywords of the hyper parameters of the lens model

- **kwargs_kin** – keyword arguments of the kinematic model hyper parameters
- **kwargs_source** – keyword argument of the source model (such as SNe)

Returns
log likelihood of the data given the model

log_likelihood_single(*ddt*, *dd*, *delta_lum_dist*, *kwargs_lens*, *kwargs_kin*, *kwargs_source*, *sigma_v_sys_error=None*)

log likelihood of the data of a lens given a specific model (as a draw from hyper-parameters) and cosmological distances

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **delta_lum_dist** – relative luminosity distance to pivot redshift
- **kwargs_lens** – keywords of the hyper parameters of the lens model
- **kwargs_kin** – keyword arguments of the kinematic model hyper parameters
- **kwargs_source** – keyword arguments of source brightness
- **sigma_v_sys_error** – unaccounted uncertainty in the velocity dispersion measurement

Returns
log likelihood given the single lens analysis for a single (random) realization of the hyper parameter distribution

luminosity_distance_modulus(*cosmo*, *z_apparent_m_anchor*)

the difference in luminosity distance between a pivot redshift (z_apparent_m_anchor) and the source redshift (effectively the ratio as this is the magnitude transform)

Parameters

- **cosmo** – astropy.cosmology instance (or equivalent with interpolation)
- **z_apparent_m_anchor** – redshift of pivot/anchor at which the apparent SNe brightness is defined relative to

Returns
lum_dist(z_source) - lum_dist(z_pivot)

sigma_v_measured_vs_predict(*cosmo*, *kwargs_lens=None*, *kwargs_kin=None*)

mean and error covariance of velocity dispersion measurement mean and error covariance of velocity dispersion predictions

Parameters

- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – keywords of the hyper parameters of the lens model
- **kwargs_kin** – keyword arguments of the kinematic model hyper parameters

Returns
sigma_v_measurement, cov_error_measurement, sigma_v_predict_mean, cov_error_predict

hierarc.Likelihood.lens_sample_likelihood module

class hierarc.Likelihood.lens_sample_likelihood.**LensSampleLikelihood**(*kwargs_lens_list*, *normalized=False*)

Bases: object

class to evaluate the likelihood of a cosmology given a sample of angular diameter posteriors Currently this class does not include possible covariances between the lens samples

log_likelihood(*cosmo*, *kwargs_lens=None*, *kwargs_kin=None*, *kwargs_source=None*)

Parameters

- **cosmo** – astropy.cosmology instance
- **kwargs_lens** – keywords of the parameters of the lens model
- **kwargs_kin** – keyword arguments of the kinematic model
- **kwargs_source** – keyword argument of the source model (such as SNe)

Returns
log likelihood of the combined lenses

num_data()
number of data points across the lens sample

Returns
integer

hierarc.Likelihood.transformed_cosmography module

class hierarc.Likelihood.transformed_cosmography.**TransformedCosmography**(*z_lens*, *z_source*)

Bases: object

class to manage hierarchical hyper-parameter that impact the cosmographic posterior interpretation of individual lenses.

displace_prediction(*ddt*, *dd*, *gamma_ppn=1*, *lambda_mst=1*, *kappa_ext=0*, *mag_source=0*)
here we effectively change the posteriors of the lens, but rather than changing the instance of the KDE we displace the predicted angular diameter distances in the opposite direction The displacements form different effects are multiplicative and thus invariant under the order those displacements are applied.

Parameters

- **ddt** – time-delay distance
- **dd** – angular diameter distance to the deflector
- **lambda_mst** – overall global mass-sheet transform applied on the sample, lambda_mst=1 corresponds to the input model
- **gamma_ppn** – post-newtonian gravity parameter (=1 is GR)
- **kappa_ext** – external convergence to be added on top of the D_dt posterior
- **mag_source** – source magnitude (attention, log scale, thus transform needs to be changed!)

Returns
ddt, dd, mag_source

Module contents

hierarc.Sampling package

Subpackages

hierarc.Sampling.ParamManager package

Submodules

hierarc.Sampling.ParamManager.cosmo_param module

class hierarc.Sampling.ParamManager.cosmo_param.**CosmoParam**(*cosmology*, *ppn_sampling=False*, *kwargs_fixed=None*)

Bases: `object`

manages the cosmological parameters in the sampling

args2kwargs(*args*, *i=0*)

Parameters
args – sampling argument list

Returns
keyword argument list with parameter names

cosmo(*kwargs*)

Parameters
kwargs – keyword arguments of parameters (can include others not used for the cosmology)

Returns
astropy.cosmology instance

kwargs2args(*kwargs*)

Parameters
kwargs – keyword argument list of parameters

Returns
sampling argument list in specified order

param_list(*latex_style=False*)

Parameters
latex_style – bool, if True returns strings in latex symbols, else in the convention of the sampler

Returns
list of the free parameters being sampled in the same order as the sampling

hierarc.Sampling.ParamManager.kin_param module

class hierarc.Sampling.ParamManager.kin_param.**KinParam**(*anisotropy_sampling=False*, *anisotropy_model='OM'*, *distribution_function='NONE'*, *sigma_v_systematics=False*, *log_scatter=False*, *kwargs_fixed=None*)

Bases: object

manager for the kinematics anisotropy parameters

args2kwargs(*args*, *i=0*)

Parameters

- **args** – sampling argument list
- **i** – integer, index to start reading out the argument list

Returns
keyword argument list with parameter names

kwargs2args(*kwargs*)

Parameters
kwargs – keyword argument list of parameters

Returns
sampling argument list in specified order

param_list(*latex_style=False*)

Parameters

- **latex_style** – bool, if True returns strings in latex symbols, else in the convention of the sampler
- **i** – int, index of the parameter to start with

Returns
list of the free parameters being sampled in the same order as the sampling

hierarc.Sampling.ParamManager.lens_param module

class hierarc.Sampling.ParamManager.lens_param.**LensParam**(*lambda_mst_sampling=False*, *lambda_mst_distribution='NONE'*, *kappa_ext_sampling=False*, *kappa_ext_distribution='NONE'*, *lambda_ifu_sampling=False*, *lambda_ifu_distribution='NONE'*, *alpha_lambda_sampling=False*, *kwargs_fixed=None*, *log_scatter=False*)

Bases: object

manages the lens model covariant parameters

args2kwargs(*args*, *i=0*)

Parameters
args – sampling argument list

Returns
keyword argument list with parameter names

kwargs2args(*kwargs*)

Parameters
kwargs – keyword argument list of parameters

Returns
sampling argument list in specified order

param_list(*latex_style=False*)

Parameters
latex_style – bool, if True returns strings in latex symbols, else in the convention of the sampler

Returns
list of the free parameters being sampled in the same order as the sampling

hierarc.Sampling.ParamManager.param_manager module

class hierarc.Sampling.ParamManager.param_manager.**ParamManager**(*cosmology*, *ppn_sampling=False*, *lambda_mst_sampling=False*, *lambda_mst_distribution='NONE'*, *anisotropy_sampling=False*, *anisotropy_model='OM'*, *anisotropy_distribution='NONE'*, *kappa_ext_sampling=False*, *kappa_ext_distribution='NONE'*, *lambda_ifu_sampling=False*, *lambda_ifu_distribution='NONE'*, *alpha_lambda_sampling=False*, *sigma_v_systematics=False*, *sne_apparent_m_sampling=False*, *sne_distribution='GAUSSIAN'*, *z_apparent_m_anchor=0.1*, *log_scatter=False*, *kwargs_lower_cosmo=None*, *kwargs_upper_cosmo=None*, *kwargs_fixed_cosmo=None*, *kwargs_lower_lens=None*, *kwargs_upper_lens=None*, *kwargs_fixed_lens=None*, *kwargs_lower_kin=None*, *kwargs_upper_kin=None*, *kwargs_fixed_kin=None*, *kwargs_lower_source=None*, *kwargs_upper_source=None*, *kwargs_fixed_source=None*)

Bases: object

class for managing the parameters involved

args2kwargs(*args*)

Parameters
args – sampling argument list

Returns
keyword argument list with parameter names

cosmo(*kwargs_cosmo*)

Parameters
kwargs_cosmo – keyword arguments of parameters (can include others not used for the cosmology)

Returns
astropy.cosmology instance

kwargs2args(*kwargs_cosmo=None*, *kwargs_lens=None*, *kwargs_kin=None*, *kwargs_source=None*)

Parameters

- **kwargs_cosmo** – keyword argument list of parameters for cosmology sampling
- **kwargs_lens** – keyword argument list of parameters for lens model sampling
- **kwargs_kin** – keyword argument list of parameters for kinematic sampling
- **kwargs_source** – keyword arguments of parameters of source brightness

Returns
sampling argument list in specified order

property num_param
number of parameters being sampled

Returns
integer

property param_bounds

Returns
argument list of the hard bounds in the order of the sampling

param_list(*latex_style=False*)

Parameters
latex_style – bool, if True returns strings in latex symbols, else in the convention of the sampler

Returns
list of the free parameters being sampled in the same order as the sampling

hierarc.Sampling.ParamManager.source_param module

class hierarc.Sampling.ParamManager.source_param.**SourceParam**(*sne_apparent_m_sampling=False*, *sne_distribution='GAUSSIAN'*, *kwargs_fixed=None*, *z_apparent_m_anchor=0.1*)

Bases: object

manager for the source property parameters (currently particularly source magnitudes for SNe)

args2kwargs(*args*, *i=0*)

Parameters

- **args** – sampling argument list
- **i** – index of argument list to start reading out

Returns
keyword argument list with parameter names

kwargs2args(*kwargs*)

Parameters
kwargs – keyword argument list of parameters

Returns
sampling argument list in specified order

param_list(*latex_style=False*)

Parameters

- **latex_style** – bool, if True returns strings in latex symbols, else in the convention of the sampler
- **i** – int, index of the parameter to start with

Returns
list of the free parameters being sampled in the same order as the sampling

Module contents

Submodules

hierarc.Sampling.mcmc_sampling module

class hierarc.Sampling.mcmc_sampling.**MCMCSampler**(**args*, ***kwargs*)

Bases: object

class which executes the different sampling methods

mcmc_emcee(*n_walkers*, *n_burn*, *n_run*, *kwargs_mean_start*, *kwargs_sigma_start*, *continue_from_backend=False*, ***kwargs_emcee*)

runs the EMCEE MCMC sampling

Parameters

- **n_walkers** – number of walkers
- **n_burn** – number of iteration of burn in (not stored in the output sample
- **n_run** – number of iterations (after burn in) to be sampled
- **kwargs_mean_start** – keyword arguments of the mean starting position
- **kwargs_sigma_start** – keyword arguments of the spread in the initial particles per parameter
- **continue_from_backend** – bool, if True and 'backend' in kwargs_emcee, will continue a chain sampling from backend

- **kwargs_emcee** – keyword argument for the emcee (e.g. to specify backend)

Returns
samples of the EMCEE run

param_names(*latex_style=False*)

list of parameter names being sampled in the same order as the sampling

Parameters
latex_style – bool, if True returns strings in latex symbols, else in the convention of the sampler

Returns
list of strings

Module contents

hierarc.Util package

Submodules

hierarc.Util.distribution_util module

class hierarc.Util.distribution_util.**PDFSampling**(*bin_edges*, *pdf_array*)

Bases: object

class for approximations with a given pdf sample

draw(*n=1*)

Returns

property draw_one

Returns

hierarc.Util.distribution_util.**approx_cdf_1d**(*bin_edges*, *pdf_array*)

Parameters

- **bin_edges** – bin edges of PDF values
- **pdf_array** – pdf array of given bins (len(bin_edges)-1)

Returns
cdf, interp1d function of cdf, inverse interpolation function

hierarc.Util.ifu_util module

this file contains routines to process the data format of the SLACS IFU data set to be binned in radial annuli and for error estimates

hierarc.Util.ifu_util.**binned_dispersion**(*dispersion_map*, *weight_map*, *flux_map*, *fiber_scale*, *r_bins*)

Parameters

- **dispersion_map** – 2d array of measured velocity dispersion for each fiber
- **weight_map** – uncertainty weight of the measurement (i.e. 1/sigma**2) for each fiber

- **flux_map** – array of flux for each fiber
- **fiber_scale** – separation of the fibers (map pixels)
- **r_bins** – array, bins in radial directions

Returns
averaged velocity dispersion measurement for each radial bin and estimated uncertainty thereof

hierarc.Util.ifu_util.**binned_total**(*dispersion_map*, *weight_map_disp*, *velocity_map*, *weight_map_v*, *flux_map*, *fiber_scale*, *r_bins*)

Parameters

- **dispersion_map** – 2d array of measured velocity dispersion for each fiber
- **weight_map_disp** – uncertainty weight of the measurement in velocity dispersion (i.e. 1/sigma**2) for each fiber
- **velocity_map** – 2d array of measured velocity offset for each fiber
- **weight_map_v** – uncertainty weight of the measurement in systemic velocity (i.e. 1/sigma**2) for each fiber
- **flux_map** – array of flux for each fiber
- **fiber_scale** – separation of the fibers (map pixels)
- **r_bins** – array, bins in radial directions

Returns
sqrt(v^2 + sigma^2) averaged integrated line dispersion when averaged over azimuthal bins

hierarc.Util.ifu_util.**binned_velocity**(*velocity_map*, *weight_map*, *flux_map*, *fiber_scale*, *r_bins*)

Parameters

- **velocity_map** – 2d array of measured velocity offset for each fiber
- **weight_map** – uncertainty weight of the velocity measurement (i.e. 1/sigma**2) for each fiber
- **flux_map** – array of flux for each fiber
- **fiber_scale** – separation of the fibers (map pixels)
- **r_bins** – array, bins in radial directions

Returns
average velocity components with luminosity weights, weight on v^2 error

hierarc.Util.likelihood_util module

hierarc.Util.likelihood_util.**cov_error_create**(*error_independent*, *error_covariance*)

generates an error covariance matrix from a set of independent uncertainties combined with a fully covariant term

Parameters

- **error_independent** – array of Gaussian 1-sigma uncertainties
- **error_covariance** – float, shared covariant error among all data points. So if all data points are off by 1-sigma, then the log likelihood is 1-sigma

Returns
error covariance matrix

hierarc.Util.likelihood_util.**get_truncated_normal**(*mean=0*, *sd=1*, *low=0*, *upp=10*, *size=1*)

Parameters

- **mean** – mean of normal distribution
- **sd** – standard deviation
- **low** – lower bound
- **upp** – upper bound

Returns
float, draw of distribution

hierarc.Util.likelihood_util.**log_likelihood_cov**(*data*, *model*, *cov_error*)
log likelihood of the data given a model

Parameters

- **data** – data vector
- **model** – model vector
- **cov_error** – inverse covariance matrix

Returns
log likelihood

Module contents

Module contents

Top-level package for hierArc.

4.4 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.4.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/sibirrer/hierarc/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

hierArc could always use more documentation, whether as part of the official hierArc docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/sibirrer/hierarc/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.4.2 Get Started!

Ready to contribute? Here's how to set up *hierarc* for local development.

1. Fork the *hierarc* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/hierarc.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv hierarc
$ cd hierarc/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 hierarc tests
$ python setup.py test or pytest
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/sibirrer/hierarc/pull_requests and make sure that the tests pass for all supported Python versions.

4.4.4 Tips

To run a subset of tests:

```
$ pytest tests.test_hierarc
```

4.4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

4.5 Credits

4.5.1 Development Lead

- Simon Birrer <sibirrer@gmail.com>

4.5.2 Contributors

- Ji Won Park jiwoncpark
- Aymeric Galan aymgal
- Martin Millon martin-millon

4.6 History

4.6.1 0.1.0 (2020-02-05)

- First release on PyPI.

4.6.2 1.0.0 (2020-06-29)

- First stable release.

4.6.3 1.1.0 (2021-07-26)

- Standardizable magnifications added

4.6.4 1.1.1 (2021-12-29)

- using CosmoInterp module from lenstronomy

4.7 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

hierarc, 43
hierarc.Diagnostics, 11
hierarc.Diagnostics.goodness_of_fit, 10
hierarc.LensPosterior, 16
hierarc.LensPosterior.anisotropy_config, 11
hierarc.LensPosterior.base_config, 12
hierarc.LensPosterior.ddt_kin_constraints, 12
hierarc.LensPosterior.ddt_kin_gauss_constraints, 14
hierarc.LensPosterior.imaging_constraints, 15
hierarc.LensPosterior.kin_constraints, 15
hierarc.Likelihood, 36
hierarc.Likelihood.anisotropy_scaling, 30
hierarc.Likelihood.cosmo_likelihood, 31
hierarc.Likelihood.hierarchy_likelihood, 32
hierarc.Likelihood.lens_sample_likelihood, 35
hierarc.Likelihood.LensLikelihood, 28
hierarc.Likelihood.LensLikelihood.base_lens_likelihood, 16
hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood, 18
hierarc.Likelihood.LensLikelihood.ddt_dd_kde_likelihood, 18
hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood, 19
hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood, 20
hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood, 21
hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood, 22
hierarc.Likelihood.LensLikelihood.ddt_lognorm_likelihood, 23
hierarc.Likelihood.LensLikelihood.ds_dds_gauss_likelihood, 24
hierarc.Likelihood.LensLikelihood.kin_likelihood, 24
hierarc.Likelihood.LensLikelihood.mag_likelihood, 26
hierarc.Likelihood.LensLikelihood.td_mag_likelihood, 27
hierarc.Likelihood.LensLikelihood.td_mag_magnitude_likelih 27
hierarc.Likelihood.SneLikelihood, 30
hierarc.Likelihood.SneLikelihood.sne_likelihood, 28
hierarc.Likelihood.SneLikelihood.sne_likelihood_custom, 28
hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file, 29
hierarc.Likelihood.SneLikelihood.sne_pantheon_plus, 30
hierarc.Likelihood.transformed_cosmography, 35
hierarc.Sampling, 41
hierarc.Sampling.mcmc_sampling, 40
hierarc.Sampling.ParamManager, 40
hierarc.Sampling.ParamManager.cosmo_param, 36
hierarc.Sampling.ParamManager.kin_param, 37
hierarc.Sampling.ParamManager.lens_param, 37
hierarc.Sampling.ParamManager.param_manager, 38
hierarc.Sampling.ParamManager.source_param, 39
hierarc.Util, 43
hierarc.Util.distribution_util, 41
hierarc.Util.ifu_util, 41
hierarc.Util.likelihood_util, 42

INDEX

A

angular_diameter_distances() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 32
ani_param_array (*hierarc.LensPosterior.anisotropy_config.AnisotropyConfig property*), 12
ani_scaling() (*hierarc.Likelihood.anisotropy_scaling.AnisotropyScalingIFU method*), 30
ani_scaling() (*hierarc.Likelihood.anisotropy_scaling.AnisotropyScalingSingleAperture method*), 30
anisotropy_kwargs() (*hierarc.LensPosterior.anisotropy_config.AnisotropyConfig method*), 12
anisotropy_scaling() (*hierarc.LensPosterior.kin_constraints.KinConstraints method*), 15
AnisotropyConfig (*class in hierarc.LensPosterior.anisotropy_config*), 11
AnisotropyScalingIFU (*class in hierarc.Likelihood.anisotropy_scaling*), 30
AnisotropyScalingSingleAperture (*class in hierarc.Likelihood.anisotropy_scaling*), 30
approx_cdf_1d() (*in module hierarc.Util.distribution_util*), 41
args2kwargs() (*hierarc.Sampling.ParamManager.cosmo_param.CosmoParam method*), 36
args2kwargs() (*hierarc.Sampling.ParamManager.kin_param.KinParam method*), 37
args2kwargs() (*hierarc.Sampling.ParamManager.lens_param.LensParam method*), 37
args2kwargs() (*hierarc.Sampling.ParamManager.param_manager.ParamManager method*), 38
args2kwargs() (*hierarc.Sampling.ParamManager.source_param.SourceParam method*), 39

B

BaseLensConfig (*class in hierarc.LensPosterior.base_config*), 12
binned_dispersion() (*in module hierarc.Util.ifu_util*), 41
binned_total() (*in module hierarc.Util.ifu_util*), 42
binned_velocity() (*in module hierarc.Util.ifu_util*), 42
build_covariance() (*hierarc.Likelihood.SneLikelihood.sne_pantheon_plus.PantheonPlusD method*), 30

C

check_list() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 32
cosmo() (*hierarc.Sampling.ParamManager.cosmo_param.CosmoParam method*), 36
cosmo() (*hierarc.Sampling.ParamManager.param_manager.ParamManage method*), 39
cosmo_instance() (*hierarc.Likelihood.cosmo_likelihood.CosmoLikelihood method*), 31
CosmoLikelihood (*class in hierarc.Likelihood.cosmo_likelihood*), 31
CosmoParam (*class in hierarc.Sampling.ParamManager.cosmo_param*), 36
cov_error_create() (*in module hierarc.Util.likelihood_util*), 42
cov_error_measurement() (*hierarc.Likelihood.LensLikelihood.kin_likelihood.KinLikelihood method*), 24
cov_error_model() (*hierarc.Likelihood.LensLikelihood.kin_likelihood.KinLikelihood method*), 25
CustomSneLikelihood (*class in hierarc.Likelihood.SneLikelihood.sne_likelihood_custom*), 28

D

ddt_dd_model_prediction() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 32
ddt_measurement() (*hierarc.Likelihood.LensLikelihood.base_lens_likelihood.LensLikeliho method*), 17
ddt_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood.DdtDdG*

method), 18
ddt_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood.DdtGaussKinLikelihood method*), 19
ddt_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood.DdtGaussianLikelihood method*), 20
ddt_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood.DdtHistKinLikelihood method*), 21
ddt_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood.DdtHistKDELikelihood method*), 22
ddt_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood.DdtHistLikelihood method*), 23
DdtDdGaussian (*class in hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood*), 18
DdtDdKDELikelihood (*class in hierarc.Likelihood.LensLikelihood.ddt_dd_kde_likelihood*), 18
DdtGaussianLikelihood (*class in hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood*), 20
DdtGaussKinConstraints (*class in hierarc.LensPosterior.ddt_kin_gauss_constraints*), 14
DdtGaussKinLikelihood (*class in hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood*), 19
DdtHistKDELikelihood (*class in hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood*), 22
DdtHistKinLikelihood (*class in hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood*), 21
DdtHistLikelihood (*class in hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood*), 22
DdtKinConstraints (*class in hierarc.LensPosterior.ddt_kin_constraints*), 12
DdtLogNormLikelihood (*class in hierarc.Likelihood.LensLikelihood.ddt_lognorm_likelihood*), 23
displace_prediction() (*hierarc.Likelihood.transformed_cosmography.TransformedCosmography method*), 35
draw() (*hierarc.Util.distribution_util.PDFSampling method*), 41
draw_anisotropy() (*hierarc.Likelihood.anisotropy_scaling.AnisotropyScalingIFU method*), 30
draw_lens() (*hierarc.LensPosterior.imaging_constraints.ImageModelPosterior method*), 15
draw_lens() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 37
draw_one (*hierarc.Util.distribution_util.PDFSampling property*), 41
draw_source() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood static method*), 33
DsDdsGaussianLikelihood (*class in hierarc.Likelihood.LensLikelihood.ds_dds_gauss_likelihood*), 24

E

error_cov_measurement (*hierarc.LensPosterior.kin_constraints.KinConstraints property*), 15

G

get_truncated_normal() (*in module hierarc.Util.likelihood_util*), 43
GoodnessOfFit (*class in hierarc.Diagnostics.goodness_of_fit*), 10

H

hierarc
    module, 43
hierarc.Diagnostics
    module, 11
hierarc.Diagnostics.goodness_of_fit
    module, 10
hierarc.LensPosterior
    module, 16
hierarc.LensPosterior.anisotropy_config
    module, 11
hierarc.LensPosterior.base_config
    module, 12
hierarc.LensPosterior.ddt_kin_constraints
    module, 12
hierarc.LensPosterior.ddt_kin_gauss_constraints
    module, 14
hierarc.LensPosterior.imaging_constraints
    module, 15
hierarc.LensPosterior.kin_constraints
    module, 15
hierarc.Likelihood
    module, 36
hierarc.Likelihood.anisotropy_scaling
    module, 30
hierarc.Likelihood.cosmo_likelihood
    module, 31
hierarc.Likelihood.hierarchy_likelihood
    module, 32
hierarc.Likelihood.lens_sample_likelihood
    module, 35
hierarc.Likelihood.LensLikelihood

module, 28
hierarc.Likelihood.LensLikelihood.base_lens_likelihood
module, 16
hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood
module, 18
hierarc.Likelihood.LensLikelihood.ddt_dd_kde_likelihood
module, 18
hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood
module, 19
hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood
module, 20
hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood
module, 21
hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood
module, 22
hierarc.Likelihood.LensLikelihood.ddt_lognorm_likelihood
module, 23
hierarc.Likelihood.LensLikelihood.ds_dds_gauss_likelihood
module, 24
hierarc.Likelihood.LensLikelihood.kin_likelihood
module, 24
hierarc.Likelihood.LensLikelihood.mag_likelihood
module, 26
hierarc.Likelihood.LensLikelihood.td_mag_likelihood
module, 27
hierarc.Likelihood.LensLikelihood.td_mag_magnitude_likelihood
module, 27
hierarc.Likelihood.SneLikelihood
module, 30
hierarc.Likelihood.SneLikelihood.sne_likelihood
module, 28
hierarc.Likelihood.SneLikelihood.sne_likelihood_custom
module, 28
hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file
module, 29
hierarc.Likelihood.SneLikelihood.sne_pantheon_plus
module, 30
hierarc.Likelihood.transformed_cosmography
module, 35
hierarc.Sampling
module, 41
hierarc.Sampling.mcmc_sampling
module, 40
hierarc.Sampling.ParamManager
module, 40
hierarc.Sampling.ParamManager.cosmo_param
module, 36
hierarc.Sampling.ParamManager.kin_param
module, 37
hierarc.Sampling.ParamManager.lens_param
module, 37
hierarc.Sampling.ParamManager.param_manager
module, 38
hierarc.Sampling.ParamManager.source_param
module, 39
hierarc.Util
module, 43
hierarc.Util.distribution_util
module, 41
hierarc.Util.ifu_util
module, 41
hierarc.Util.likelihood_util
module, 42
hierarchy_configuration() (*hierarc.LensPosterior.ddt_kin_constraints.DdtKinConstraints method*), 13
hierarchy_configuration() (*hierarc.LensPosterior.ddt_kin_gauss_constraints.DdtGaussKinConstr method*), 14
hierarchy_configuration() (*hierarc.LensPosterior.kin_constraints.KinConstraints method*), 15
hyper_param_likelihood() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 33

I

ImageModelPosterior (*class in hierarc.LensPosterior.imaging_constraints*), 15

J

j_kin_draw() (*hierarc.LensPosterior.kin_constraints.KinConstraints method*), 16

K

kin_fit() (*hierarc.Diagnostics.goodness_of_fit.GoodnessOfFit method*), 10
KinConstraints (*class in hierarc.LensPosterior.kin_constraints*), 15
KinLikelihood (*class in hierarc.Likelihood.LensLikelihood.kin_likelihood*), 24
KinParam (*class in hierarc.Sampling.ParamManager.kin_param*), 37
kwargs2args() (*hierarc.Sampling.ParamManager.cosmo_param.CosmoP method*), 36
kwargs2args() (*hierarc.Sampling.ParamManager.kin_param.KinParam method*), 37
kwargs2args() (*hierarc.Sampling.ParamManager.lens_param.LensParam method*), 38
kwargs2args() (*hierarc.Sampling.ParamManager.param_manager.Param method*), 39
kwargs2args() (*hierarc.Sampling.ParamManager.source_param.SourceP method*), 40

kwargs_anisotropy_base (*hierarc.LensPosterior.anisotropy_config.AnisotropyConfig property*), 12

L

lens_log_likelihood() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 33
LensLikelihood (*class in hierarc.Likelihood.hierarchy_likelihood*), 32
LensLikelihoodBase (*class in hierarc.Likelihood.LensLikelihood.base_lens_likelihood*), 16
LensParam (*class in hierarc.Sampling.ParamManager.lens_param*), 37
LensSampleLikelihood (*class in hierarc.Likelihood.lens_sample_likelihood*), 35
likelihood() (*hierarc.Likelihood.cosmo_likelihood.CosmoLikelihood method*), 31
log_likelihood() (*hierarc.Likelihood.lens_sample_likelihood.LensSampleLikelihood method*), 35
log_likelihood() (*hierarc.Likelihood.LensLikelihood.base_lens_likelihood.LensLikelihoodBase method*), 17
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood.DdtDdGaussian method*), 18
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_dd_kde_likelihood.DdtDdKDELikelihood method*), 18
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood.DdtGaussKinLikelihood method*), 19
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood.DdtGaussianLikelihood method*), 20
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood.DdtHistKinLikelihood method*), 21
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood.DdtHistLikelihood method*), 22
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood.DdtHistLikelihood method*), 23
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ddt_lognorm_likelihood.DdtLogNormLikelihood method*), 23
log_likelihood() (*hierarc.Likelihood.LensLikelihood.ds_dds_gauss_likelihood.DsDdsGaussianLikelihood method*), 24
log_likelihood() (*hierarc.Likelihood.LensLikelihood.kin_likelihood.KinLikelihood method*), 25
log_likelihood() (*hierarc.Likelihood.LensLikelihood.mag_likelihood.MagnificationLike method*), 26
log_likelihood() (*hierarc.Likelihood.LensLikelihood.td_mag_likelihood.TDMagLikeli method*), 27
log_likelihood() (*hierarc.Likelihood.LensLikelihood.td_mag_magnitude_likelihood.TD method*), 27
log_likelihood() (*hierarc.Likelihood.SneLikelihood.sne_likelihood.SneLikelihood method*), 28
log_likelihood_cov() (*in module hierarc.Util.likelihood_util*), 43
log_likelihood_lum_dist() (*hierarc.Likelihood.SneLikelihood.sne_likelihood_custom.CustomSneL method*), 28
log_likelihood_lum_dist() (*hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file.SneLikeli method*), 29
log_likelihood_single() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 34
luminosity_distance_modulus() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 34

M

MagDdDdtKDELikelihood (*class in hierarc.Likelihood.LensLikelihood.mag_likelihood*), 26
make_sampler() (*hierarc.Sampling.mcmc_sampling.MCMCSampler method*), 40
MCMCSampler (*class in hierarc.Sampling.mcmc_sampling*), 40
model_marginalization() (*hierarc.LensPosterior.kin_constraints.KinConstraints method*), 16
module
    hierarc, 43
    hierarc.Diagnostics, 11
    hierarc.Diagnostics.goodness_of_fit, 10
    hierarc.LensPosterior, 16
    hierarc.LensPosterior.anisotropy_config, 11
    hierarc.LensPosterior.base_config, 12
    hierarc.LensPosterior.ddt_kin_constraints, 12
    hierarc.LensPosterior.ddt_kin_gauss_constraints,

hierarc.LensPosterior.imaging_constraints, 15
hierarc.LensPosterior.kin_constraints, 15
hierarc.Likelihood, 36
hierarc.Likelihood.anisotropy_scaling, 30
hierarc.Likelihood.cosmo_likelihood, 31
hierarc.Likelihood.hierarchy_likelihood, 32
hierarc.Likelihood.lens_sample_likelihood, 35
hierarc.Likelihood.LensLikelihood, 28
hierarc.Likelihood.LensLikelihood.base_lens_likelihood, 16
hierarc.Likelihood.LensLikelihood.ddt_dd_gauss_likelihood, 18
hierarc.Likelihood.LensLikelihood.ddt_dd_kde_likelihood, 18
hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood, 19
hierarc.Likelihood.LensLikelihood.ddt_gauss_likelihood, 20
hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood, 21
hierarc.Likelihood.LensLikelihood.ddt_hist_likelihood, 22
hierarc.Likelihood.LensLikelihood.ddt_lognorm_likelihood, 23
hierarc.Likelihood.LensLikelihood.ds_dds_gauss_likelihood, 24
hierarc.Likelihood.LensLikelihood.kin_likelihood, 24
hierarc.Likelihood.LensLikelihood.mag_likelihood, 26
hierarc.Likelihood.LensLikelihood.td_mag_likelihood, 27
hierarc.Likelihood.LensLikelihood.td_mag_magnitude_likelihood, 27
hierarc.Likelihood.SneLikelihood, 30
hierarc.Likelihood.SneLikelihood.sne_likelihood, 28
hierarc.Likelihood.SneLikelihood.sne_likelihood_custom, 28
hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file, 29
hierarc.Likelihood.SneLikelihood.sne_pantheon_plus, 30
hierarc.Likelihood.transformed_cosmography, 35
hierarc.Sampling, 41
hierarc.Sampling.mcmc_sampling, 40
hierarc.Sampling.ParamManager, 40
hierarc.Sampling.ParamManager.cosmo_param, 36
hierarc.Sampling.ParamManager.kin_param, 37
hierarc.Sampling.ParamManager.lens_param, 37
hierarc.Sampling.ParamManager.param_manager, 38
hierarc.Sampling.ParamManager.source_param, 39
hierarc.Util, 43
hierarc.Util.distribution_util, 41
hierarc.Util.ifu_util, 41
hierarc.Util.likelihood_util, 42

N

num_data() (*hierarc.Likelihood.lens_sample_likelihood.LensSampleLikeli method*), 35
num_data() (*hierarc.Likelihood.LensLikelihood.base_lens_likelihood.Lens method*), 17
num_param (*hierarc.Sampling.ParamManager.param_manager.ParamMana property*), 39

P

PantheonPlusData (*class in hierarc.Likelihood.SneLikelihood.sne_pantheon_plus*), 30
param_bounds (*hierarc.Sampling.ParamManager.param_manager.ParamM property*), 39
param_list() (*hierarc.Sampling.ParamManager.cosmo_param.CosmoPa method*), 36
param_list() (*hierarc.Sampling.ParamManager.kin_param.KinParam method*), 37
param_list() (*hierarc.Sampling.ParamManager.lens_param.LensParam method*), 38
param_list() (*hierarc.Sampling.ParamManager.param_manager.ParamM method*), 39
param_list() (*hierarc.Sampling.ParamManager.source_param.SourcePa method*), 40
param_names() (*hierarc.Sampling.mcmc_sampling.MCMCSampler method*), 41
ParamManager (*class in hierarc.Sampling.ParamManager.param_manager*), 38
PDFSampling (*class in hierarc.Util.distribution_util*), 41
plot_adt_fit() (*hierarc.Diagnostics.goodness_of_fit.GoodnessOfFit method*), 10
plot_ifu_fit() (*hierarc.Diagnostics.goodness_of_fit.GoodnessOfFit method*), 11
plot_kin_fit() (*hierarc.Diagnostics.goodness_of_fit.GoodnessOfFit method*), 11

R

read_covariance_matrix() (*in module hier-*

arc.Likelihood.SneLikelihood.sne_likelihood_from_file), 29
reduced_chi2() (*hierarc.Diagnostics.goodness_of_fit.GoodnessOfFit method*), 11

S

sigma_v_measured_vs_predict() (*hierarc.Likelihood.hierarchy_likelihood.LensLikelihood method*), 34
sigma_v_measurement() (*hierarc.Likelihood.LensLikelihood.base_lens_likelihood.LensLikelihoodBase method*), 17
sigma_v_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood.DdtGaussKinLikelihood method*), 19
sigma_v_measurement() (*hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood.DdtHistKinLikelihood method*), 21
sigma_v_measurement() (*hierarc.Likelihood.LensLikelihood.kin_likelihood.KinLikelihood method*), 25
sigma_v_measurement_mean() (*hierarc.Likelihood.LensLikelihood.kin_likelihood.KinLikelihood method*), 25
sigma_v_model() (*hierarc.Likelihood.LensLikelihood.kin_likelihood.KinLikelihood method*), 25
sigma_v_prediction() (*hierarc.Likelihood.LensLikelihood.base_lens_likelihood.LensLikelihoodBase method*), 17
sigma_v_prediction() (*hierarc.Likelihood.LensLikelihood.ddt_gauss_kin_likelihood.DdtGaussKinLikelihood method*), 20
sigma_v_prediction() (*hierarc.Likelihood.LensLikelihood.ddt_hist_kin_likelihood.DdtHistKinLikelihood method*), 21
sigma_v_prediction() (*hierarc.Likelihood.LensLikelihood.kin_likelihood.KinLikelihood method*), 26
SneLikelihood (*class in hierarc.Likelihood.SneLikelihood.sne_likelihood*), 28
SneLikelihoodFromFile (*class in hierarc.Likelihood.SneLikelihood.sne_likelihood_from_file*), 29
SourceParam (*class in hierarc.Sampling.ParamManager.source_param*), 39

T

TDMagLikelihood (*class in hierarc.Likelihood.LensLikelihood.td_mag_likelihood*), 27
TDMagMagnitudeLikelihood (*class in hierarc.Likelihood.LensLikelihood.td_mag_magnitude_likelihood*), 27
TransformedCosmography (*class in hierarc.Likelihood.transformed_cosmography*), 35