PyThia

Release 01.02.2018

Nando Farchmin

Feb 14, 2023

CONTENTS:

CHAPTER

ONE

PYTHIA

1.1 pythia package

1.1.1 Submodules

1.1.2 pythia.basis module

Assemble sparse univariate and multivariate basis polynomials.

Build univariate or multivariate normalized basis polynomials depending on the domain and distribution (and other degrees of freedom) of the parameter(s).

Currently supported are the following distribution types:

- uniform
- normal
- Gamma
- Beta

pythia.basis.**multivariate_basis**(*paramBasis*, *mdx*, *partial=None*)

Assemble multivariate polynomial basis.

Set the (partial derivative of the) multivariate (product) polynomial basis functions.

Parameters

- **paramBasis** (*list of list of callable*) – Univariate basis functions for parameters. Is called by *paramBasis[paramIdx][deg]()*.
- **mdx** (*array_like*) – Array of multiindices for multivariate basis functions.
- **partial** (*list of int*) – Number of partial derivatives for each dimension. Length is same as *paramBasis*.

Returns

pcBasis – List of normalized multivariate polynomials w.r.t. parameter domain and distribution and univariate degrees as specified in *mdx*.

Return type

list of functions

pythia.basis.**normalize_polynomial**(*weight*, *p_list*, *param*)

Normalize orthogonal polynomials.

Normalize a polynomial of an orthogonal system with respect to the scalar product

$$a(u,v)_{\mathrm{pdf}} = \int u(p)v(p)\mathrm{pdf}(p)\mathrm{d}p.$$

The normalized polynomial ϕ_j for any given polynomial P_j is given by $\phi_j = P_j/\sqrt{c_j}$ for the constant $c_j = \int \mathrm{pdf}(p) * P_j(p)^2\mathrm{d}p$.

Parameters

- **weight** (*callable*) – Probability density function.
- **p_list** (list of *numpy.polynomial.Polynomial*) – Polynomials to normalize w.r.t. weight.
- **param** (*pythia.parameter.Parameter*) – Parameter used for distribution and domain information.

Returns
List of normalized univariate polynomials.

Return type
list of function

pythia.basis.**set_hermite_basis**(*param*, *deg*)

Generate list of probabilists Hermite polynomials.

Generate the Hermite Polynomials up to certain degree according to the mean and variance of the specified parameter.

Parameters

- **param** (*pythia.parameters.Parameter*) – Parameter for basis function. Needs to be normal distributed.
- **deg** (*int*) – Maximum degree for polynomials.

Returns
Probabilists Hermite polynomials up to (including) degree specified in *deg*.

Return type
list of function

pythia.basis.**set_jacobi_basis**(*param*, *deg*)

Generate list of Jacobi polynomials.

Generate the Jacobi Polynomials up to certain degree on the interval and DoFs specified by the parameter.

Parameters

- **param** (*pythia.parameters.Parameter*) – Parameter for basis function. Needs to be Beta-distributed.
- **deg** (*int*) – Maximum degree for polynomials.

Returns
Jacobi polynomials up to (including) degree specified in *deg*.

Return type
list of function

pythia.basis.**set_laguerre_basis**(*param*, *deg*)

Generate list of Leguerre polynomials.

Generate the generalized Laguerre polynomials up to certain degree on the interval and DoFs specified by the parameter.

Parameters

- **param** (*pythia.parameters.Parameter*) – Parameter for basis function. Needs to be Gamma-distributed.
- **deg** (*int*) – Maximum degree for polynomials.

Returns
Laguerre polynomials up to (including) degree specified in *deg*.

Return type
list of function

pythia.basis.**set_legendre_basis**(*param*, *deg*)

Generate list of the Legendre Polynomials.

Generate the Legendre Polynomials up to certain degree on the interval specified by the parameter.

Parameters

- **param** (*pythia.parameters.Parameter*) – Parameter for basis function. Needs to be uniformly distributed.
- **deg** (*int*) – Maximum degree for polynomials.

Returns
Legendre polynomials up to (including) degree specified in *deg*.

Return type
list of function

pythia.basis.**univariate_basis**(*params*, *degList*)

Assemble a univariate polynomial basis.

Set polynomial basis up to deg for each parameter in *params* according to the parameter distribution and area of definition.

Parameters

- **params** (list of *pythia.parameter.Parameter*) – Parameters to compute univariate basis function for.
- **degList** (*array_like*) – Max. degrees of univariate polynomials for each parameter.

Returns
basis – List of normalized univariate polynomials w.r.t. parameter domain and distribution up to specified degree for each parameter in *params*.

Return type
list of list of functions

1.1.3 pythia.chaos module

Sample-based computation of polynomial chaos expansion.

This module provides classes to compute the polynomial chaos approximation of an unknown function given via input/output training data pairs via linear least-squares regression.

class pythia.chaos.**PolynomialChaos**(*params*, *mdx*, *regPoints*, *weights*, *fEval*, *xVal=None*, *infoMatrix=None*, *basisMat=None*, *regLength=0*, *paramPDF=None*, *pdf=None*, *paramBasis=None*, *basis=None*, *coeff=None*, *verbose=False*)

Bases: `object`

Computation of sparse polynomial chaos expansion.

Parameters

- **params** (dict/list of *pythia.parameter.RandomParameter*) – Parameter to specify random variables for chaos expansion.
- **mdx** (*pythia.index.PCMultiIndex*) – Multiindices of coefficients of the polynomial chaos expansion.
- **regPoints** (*array_like*) – Regression points for linar regression (Parameter realizations).
- **weights** (*array_like*) – Regression weights for *regPoints*.
- **fEval** (*array_like*) – Function evaluation of *regPoints* and *xVal*.
- **xVal** (*array_like, optional*) – Spacial/physical point of function evaluation.
- **infoMatrix** (*array_like, optional*) – Information matrix (Gramian) of the basis functions and samples.
- **basisMat** (*array_like, optional*) – Basis evaluation matrix of samples.
- **regLength** (*int, default=0*) – Number of regression points used in linear regression. Needs to be less or equal to length of mdx. Zero results in automatic choice of amount of sample points.
- **paramPDF** (*list of function, optional*) – List of univariate PDFs for each parameter.
- **pdf** (*function, optional*) – Multivariate PDF of parameters.
- **paramBasis** (*list of list of function, optional*) – Univariate basis functions for each parameter.
- **basis** (*list of function, optional*) – Multivariate basis functions as specified in *mdx*.
- **coeff** (*array_like, optional*) – PC coefficients.
- **verbose** (*bool, default=False*) – Makes PCE talk.

approximation(*x*, *partial=None*)

Evaluate the (partial derivative of the) PC approximation.

Parameters

- **x** (*array_like*) – Parameter realizations in which the approximation is evaluated.
- **partial** (*list of int*) – List that specifies the number of derivatives in each component. Length is the number of parameters.

Returns
Evaluation of PC expansion in parameter realizations.

Return type
array_like

property basis

Multivariate basis functions used for the PC expansion.

property basisMat

Basis evaluation matrix $P_j(y_i)$.

property coeff

Coefficients of the PC expansion.

property fDim

Image dimension of the function.

property fEval

Function evaluations used for training.

property infoMatrix

Information matrix (unscaled Gramian) for the samples.

property mdx

Multiindices object.

property mean

Mean of the PC expansion.

property pDim

Number of random parameters.

property paramBasis

List of univariate basis functions for each parameter.

property paramPDF

List of univariate PDFs for each parameter.

property parameter

List of parameters.

property pdf

Multivariate PDF of parameters.

property regLength

Number of used training data to compute PC expansion.

property regPoints

Parameter realization used for training.

property sobolCoeff

Approximated Sobol indices for the PC expansion.

property std

Standard deviation of the PC expansion.

property variance

Variance of the PC expansion.

property weights

Regression weights.

property xDim

Spatial dimension.

property xVal

Spatial component.

class pythia.chaos.**PolynomialChaosFrame**(*params*, *mdx*, *regPoints*, *weights*, *fEval*, *xVal*, *infoMatrix=None*, *basisMat=None*, *coeff=None*, *regLength=0*, *verbose=False*)

Bases: `object`

Computation of sparse PC expansion for each spatial component.

Deprecated since version 2.0.0: *PolynomialChaosFrame* will be removed in PyThia 3.0.0, it is replaced by *PolynomialChaos* because the latter performs more efficiently.

Compute the PC expansion of a function for given parameters for specified multiindices with linear regression. The coefficients are computed for each spatial component individually.

Parameters

- **params** (dict/list of *pythia.parameter.RandomParameter*) – Parameter to specify random variables for chaos expansion.
- **mdx** (*pythia.index.PCMultiIndex*) – Multiindices of coefficients of the polynomial chaos expansion.
- **regPoints** (*array_like*) – Regression points for linar regression (Parameter realizations).
- **weights** (*array_like*) – Regression weights for *regPoints*.
- **fEval** (*array_like*) – Function evaluation of *regPoints* and *xVal*.
- **xVal** (*array_like, optional*) – Spacial/physical point of function evaluation.
- **infoMatrix** (*array_like, optional*) – Information matrix (Gramian) of the basis functions and samples.
- **basisMat** (*array_like, optional*) – Basis evaluation matrix of samples.
- **coeff** (*array_like, optional*) – PC coefficients.
- **regLength** (*int, default=0*) – Number of regression points used in linear regression. Needs to be less or equal to length of mdx. Zero results in automatic choice of amount of sample points.
- **verbose** (*bool, default=False*) – Makes PCE talk.

approximation(*x*, *partial=None*)

Evaluate the (partial derivative of the) PC approximation.

Parameters

- **x** (*array_like*) – Parameter realizations in which the approximation is evaluated.
- **partial** (*list of int*) – List that specifies the number of derivatives in each component. Length is the number of parameters.

Returns
Evaluation of PC expansion in parameter realizations.

Return type
array_like

property basis

Multivariate basis functions used for the PC expansion.

property basisMat

Basis evaluation matrix $P_j(y_i)$.

property coeff

Coefficients of the PC expansion.

property fDim

Image dimension of the function.

property fEval

Function evaluations used for training.

property infoMatrix

Information matrix (unscaled Gramian) for the samples.

property mdx

Multiindices object.

property mean

Mean of the PC expansion.

property pDim

Number of random parameters.

property paramBasis

List of univariate basis functions for each parameter.

property paramPDF

List of univariate PDFs for each parameter.

property parameter

List of parameters.

property pdf

Multivariate PDF of parameters.

property regLength

Number of used training data to compute PC expansion.

property regPoints

Parameter realization used for training.

run(*coeff=None*)

Compute PC expansion for each spatial component individually.

Compute the PC expansion coefficients for each spatial component without repeatedly assembling the information matrix.

Parameters

coeff (*array_like, optional*) – Precomputed PC coefficients.

Notes

This method needs to be called separately after initiating a *PolynomialChaosFrame* object.

property sobolCoeff

Approximated Sobol indices for the PC expansion.

property std

Standard deviation of the PC expansion.

property variance

Variance of the PC expansion.

property weights

Regression weights.

property xDim

Spatial dimension.

property xVal

Spatial component.

pythia.chaos.**assemble_mdx**(*IDXs*, *VALs*, *SDXs*, *nParams*, *max_terms*)

Compute automatic choice of multiindices.

Parameters

- **IDXs** (*list*) – Enumeration of all possible Sobol indices.
- **VALs** (*array_like*) – Values of Sobol indices.
- **SDXs** (*list*) – Sobol indices (subscripts, not the values).
- **nParams** (*int*) – Number of stochastic parameters.
- **max_terms** (*int*) – Maximum number of expansion terms.

Returns
mdx – Sparse multiindices.

Return type
pythia.index.PCMultiIndex

pythia.chaos.**find_optimal_mdx**(*params*, *regPoints*, *weights*, *fEval*, *max_terms=0*, *threshold=0.001*, *get_sobol_coeff=False*)

Compute optimal multiindices of PCE.

Compute the optimal multiindices for a polynomial chaos expansion based on an estimate of the Sobol coefficient values.

Parameters

- **params** (*list of pythia.Parameters.RandomParameter*) – Random parameters of the problem.
- **regPoints** (*array_like*) – Sample points for training
- **weights** (*array_like*) – Weights for training.
- **fEval** (*array_like*) – Function evaluations for training.
- **max_terms** (*int, default=0*) – Maximum number of expansion terms. Number of expansion terms is chosen automatically for *max_terms=0*.

- **get_sobol_coeff** (*bool*) – If true, function also return crude intermediate approximation of Sobol coefficient values.

Returns

- **mdx** (*pythia.Indices.PCMultiIndex*) – Adaptive multiindices.
- **sobolCoeff** (*array_like, optional*) – Crude intermediate approximation of Sobol indices.

pythia.chaos.**get_infoMat_batchsize**(*dim*, *save_memory=538445312.5*)

Compute memory allocation batch sizes for information matrix.

Compute the maximal number of samples in each batch when assembling the information matrix to be maximally memory efficient and avoid OutOfMemory errors.

Parameters

- **dim** (*int*) – Number of rows/columns of information matrix.
- **save_memory** (*int, default=3*1025/2*) – Memory (in bytes), that should be kept free. The default is equivalent to 512 MB.

Returns
n – Batchsize for assembling of information matrix.

Return type
int

pythia.chaos.**get_sobol_coefficients**(*params*, *regPoints*, *weights*, *fEval*, *max_terms*)

Compute rough estimate of Sobol coefficient.

Compute an approximation of the Sobol coefficients with as many PC expansion terms as possible.

Parameters

- **params** (list of *pythia.parameter.RandomParameter*) – List of stochastic parameters.
- **regPoints** (*array_like*) – Parameter realizations.
- **weights** (*array_like*) – Regressio weights.
- **fEval** (*array_like*) – Function evaluations in regression points.
- **max_terms** (*int*) – Number of considered expansion terms.

Returns

- **pc.sobolCoeff** (*array_like*) – Sobol indices.
- **mdx.sdx** (*list of tuple*) – Sobol index identifiers (subscripts).

1.1.4 pythia.density module

Assemble univariate and multivariate PDFs for stochastic parameters.

Build the univariate and multivariate probability density functions (PDFs) for *pythia.parameter.RandomParameter* parameters.

pythia.density.**multivariatePDF**(*uPDF*)

Compute the multivariate (product) PDF of the given parameters.

Parameters
uPDF (*list of function*) – List of univariate PDFs.

Returns
pdf – Multivariate parameter PDF.

Return type
function

pythia.density.**paramPDF**(*params*)

Set the PDF for each parameter according to the parameter distribution.

The PDF of parameter *j* can be evaluated at some point $x \in \mathbb{R}$ by *pdf[j](x)*.

Parameters
params (list of *pythia.parameters.RandomParameter*) – List of parameters with information of the probability distribution.

Returns
pdf – List of univariate PDFs for each parameter.

Return type
list of function

1.1.5 pythia.index module

Store information about sparse multiindices.

Provide information about sparse PC expansion terms in form of multiindices of polynomial degrees for each of the parameter dimensions. Also compute Sobol indices subscripts and provide conversions of PC multiindices (mdx) Sobol multiindices (sdx) and linear enumeration (idx).

class pythia.index.**PCMultiIndex**(*dimension*)

Bases: object

Generate multiindices for sparse PC expansion.

Parameters
dimension (*int*) – Number of stochastic parameters.

property count

Number of PC expansion terms.

property dimension

Number of stochastic parameters.

generateMDX2IDX()

Generate dict to map *tuple(mdx)* to corresponding index.

generateMDX2SDX()

Generate dictionary to map *mdx* to *sdx*.

Generate dictionary with structre *dct[tuple(mdx)] = sdx*.

Note: The *mdx* are numpy ndarrays which cannot be cast to keys in python dictionaries. Hence the individual multiindices need to be cast to tuples before using the dictionary.

---

generateSDX()

Generate Sobol index subscripts for parameter dimension.

generateSDX2IDX()

Generate dict to map *tuple(sdx)* to corresponding index.

generateSDX2MDX()

Generate dictionary to map *sdx* to all corresponding *mdx*.

property maximum

Maximum univariate polynomial degree.

property mdx

Array of PC expansion multiindices.

property mdx2idx

Dictionary to map mdx to enumeration index.

property mdx2sdx

Dictionary to map mdx to corresponding sdx.

mdxAdd(*pos*, *deg*)

Assemble array of multiindices as cartesian product of deg in pos.

Parameters

- **pos** (*list*) – Positions where multiindices are added.
- **deg** (*list of lists*) – Multiindex values for position of same length as *pos*.

Returns
Sorted array of PC multiindices.

Return type
array_like

mdxFromShape(*shape*)

Compute (full tensor) multiindices up to specified degree.

Parameters
shape (*iterable*) – Maximal indices for multiindex.

mdxLimitAbs(*limit*)

Limit maximal dimension of multiindices.

Restrict *self.mdx* to a sub-array of multiindices with total degree less or equal to the specified limit.

Parameters
limit (*int*) – Limit for the total degree of the multiindices.

mdxSort(*mdx*)

Sort mdx by sum.

Parameters
mdx (*array_like*) – Array of PC multiindices.

refresh()

Generate Sobol indices and index dictionaries from *mdx* array.

property sdx

List of Sobol indices subscripts.

property sdx2mdx

Dictionary to map sdx to all corresponding mdx.

1.1.6 pythia.likelihood module

Provide Gaussian likelihoods for PC surrogates.

class pythia.likelihood.**Gaussian**(*f*, *sigma*, *xdim*)

Bases: object

Gaussian likelihood function for differentiable forward problem.

Assemble the Gaussian likelihood

$$\mathcal{L}(x) = \frac{1}{(2\pi)^{M/2}\sqrt{\det \Sigma}} \exp\Big(-\frac{1}{2}\|\Sigma^{-1/2}(f(x) - \delta)\|\Big)$$

for covariance matrix $\Sigma = \mathrm{diag}(\sigma_1(y_1), \ldots, \sigma_M(y_M))$ and measurement/observation δ.

Parameters

- **f** (*function*) – Forward model.
- **sigma** (*function*) – Error model for standard deviation describing Σ.
- **xdim** (*int*) – Number of stochastic parameters.

likelihood(*x*, *y_meas*)

Evaluate the Gaussian likelihood with specified measurement.

Parameters

- **x** (*array_like*) – Realizations of stochastic parameters.
- **y_meas** (*array_like*) – Measurement δ.

log_likelihood(*x*, *y_meas*)

Evaluate the Gaussian log-likelihood with specified measurement.

Parameters

- **x** (*array_like*) – Realizations of stochastic parameters.
- **y_meas** (*array_like*) – Measurement δ.

pythia.likelihood.**gauss_likelihood**(*p*, *y*, *error_model*)

Compute a gaussian likelihood function.

Deprecated since version 2.0.0: *gauss_likelihood* will be removed in PyThia 3.0.0, it is replaced by the *Gaussian* class as the latter is clearer and more comfortable to use.

Parameters

- **p** (*array_like*) – Parameter realizations.
- **y** (*array_like*) – Measurement data δ.
- **error_model** (*function*) – Function that maps *p* to two arrays containing the evaluation of the forward model *f(p)* and the standard deviation *sigma(p)*.

1.1.7 pythia.loggers module

Logger class to write output to terminal and in log file.

class pythia.loggers.**Logger**(*fileName='logfile.log'*, *filePath='./Log/'*)

Bases: object

Logger to write function output to terminal and in log file.

Parameters

- **fileName** (*string, default='logfile.log'*) – Name of the log file.
- **filePath** (*string, default='./Log/'*) – Path to the log file.

Examples

```
>>> sys.stdout = Logger(fileName, filePath)
```

flush()

Flush screen.

write(*message*)

Write string to terminal and log file.

pythia.loggers.**log_progress**(*sequence*, *every=None*, *size=None*, *name='Items'*)

Progress bar for jupyter notebooks.

Deprecated since version 2.0.0: *log_progress* will be removed in PyThia 3.0.0 as this is not part of the PyThia core functionality.

Note: Code copied from https://github.com/alexanderkuk/log-progress

1.1.8 pythia.mcmc_sampling module

pythia.mcmc_sampling.**random_walk_metropolis**(*density*, *domain*, *seeds*, *CHAIN_LENGTH*, *DELTA*)

Random walk Metropolis-Hastings algorithm for sampling densities.

Since the random walks for multiple seeds are computed in parallel, the seed length of each seed is chosen uniformly.

Deprecated since version 2.0.0: This is just for benchmarking MCMC sampling. Use other python libraries for reasonable computations.

Parameters

- **density** (*function*) – Density (not necessarily normalized) from which the samples are generated.
- **seeds** (*array_like, ndim = 2*) – Seed for stochastic sampling. First dimension is the number of seeds used for parallel sampling, second dimension is the dimension of the samples/parameters.
- **CHAIN_LENGTH** (*int*) – Length of the Markov chains.
- **DELTA** (*array_like*) – Standart deviation of noise direction added in each step for each parameter.

1.1.9 pythia.misc module

pythia.misc.**batch**(*iterable*, *n=1*)

Split iterable into different batches of batchsize n.

Parameters

- **iterable** (*array_like*) – Iterable to split.
- **n** (*int, default=1*) – Batch size.

Yields
iterable – Iterable for different batches.

pythia.misc.**cartProd**(*arrayList*)

Compute the outer product of two or more arrays.

Assemble an array containing all possible combinations of the elements of the input vectors $v_1, \ldots, v_n$.

Parameters
arrayList (*list of array_like*) – List of vectors $v_1, \ldots, v_n$.

Returns
x – Cartesian product array.

Return type
array_like

pythia.misc.**distributionDict**()

Set aliases for distribution descriptions.

Deprecated since version 2.0.0: *distributionDict* will be removed in PyThia 3.0.0.

Returns
distDict – Dictionary with aliases for distribution descriptions.

Return type
dict

pythia.misc.**doerfler_marking**(*values*, *idx=None*, *threshold=0.9*)

Dörfler marking for arbitrary values.

Parameters

- **values** (*array_like*) – Values for the Dörfler marking.
- **idx** (*list of int, optional*) – List of indices associated with the entries of *values*. If *None*, this is set to `range(len(values))`.
- **threshold** (*float, default=0.9*) – Threshold paramter for Dörfler marking.

Returns

- **idx_reordered** (*list*) – Reordered indices given by *idx*. Ordered from largest to smallest value.
- **ordered_values** (*array_like*) – Reordered values. Ordered from largest to smallest.
- **marker** (*int*) – Threshold marker such that `sum(values[:marker]) > threshold * sum(values)`.

pythia.misc.**formatTime**(*dt*)

Converts time (seconds) to time format string.

Parameters
dt (*float*) – Time in seconds.

Returns
Formatted time string.

Return type
str

pythia.misc.**gelman_rubin_condition**(*chains*)

Compute Gelman-Rubin criterion.

Implementation of the Gelman-Rubin convergence criterion for multiple parameters. A Markov chain is said to be in its convergence, if the final ration is close to one.

Parameters
chains (*array_like, ndim=3*) – Array containing the Markov chains of each parameter. All chains are equal in length, the assumed shape is `(#chains, chain length, #params)`.

Returns
Value computed by Gelman-Rubin criterion.

Return type
float

pythia.misc.**get_confidence_interval**(*samples*, *rate=0.95*, *resolution=500*)

Compute confidence intervals of samples.

Compute the confidence intervals of the 1D marginals of the samples (slices). The confidence interval of a given rate is the interval around the median (not mean) of the samples containing roughly *rate* percent of the total mass. This is computed for the left and right side of the median independently.

Parameters

- **samples** (*array_like, ndim < 3*) – Array containing the (multidimensional) samples.
- **rate** (*float, default=0.95*) – Fraction of the total mass the interval should contain.
- **resolution** (*int, default=500*) – Number of bins used in histogramming the samples.

Returns
conf_intervals – Confidence intervals for each component.

Return type
array_like

pythia.misc.**is_contained**(*val*, *domain*)

Check if a given value (vector) is contained in a domain.

Checks if each component of the vector lies in the one dimensional interval of the corresponding component of the domain.

Parameters

- **val** (*array_like*) – Vector to check containment in domain
- **domain** (*array_like*) – Product domain of one dimensional intervals.

Returns
Bool stating if value is contained in domain.

Return type
bool

pythia.misc.**line**(*indicator*, *message=None*)

Print a line of 80 characters by repeating indicator.

An additional message can be given.

Parameters

- **indicator** (*string*) – Indicator the line consists of, e.g. '-', '+' or '+-'.
- **message** (*string, optional*) – Message integrated in the line.

Returns
text – String of 80 characters length.

Return type
str

pythia.misc.**load**(*filename*)

Alias for numpy.load().

pythia.misc.**now**(*sep=False*)

Get string of current machine date and time.

Parameters
sep (*bool, default=False*) – If true, return date and time separately.

Returns
Tuple containing date and time or single string of both concatenated.

Return type
tuple of str or str

pythia.misc.**paramDictToList**(*paramDict*)

Cast dictionary of pythia parameters to a list sorted by index.

Deprecated since version 2.0.0: *paramDictToList* will be removed in PyThia 3.0.0, parameters should always be given as a list not a dictionary.

Parameters
paramDict (*dict*) – Dictionary of parameter names and corresponding *pythia.parameter.RandomParameter*.

Returns
paramList – Parameter list sorted by index of parameter.

Return type
list of *pythia.parameter.RandomParameter*

pythia.misc.**save**(*filename*, *data*, *path='./'*)

Wrapper for *numpy.save* that assures path directory is created if necessary and that backs old data up if existent.

Parameters

- **name** (*str*) – Filename to save data to.
- **data** (*array_like*) – Data to save as .npy file.
- **path** (*str, default='./'*) – Path under which the file should be created.

pythia.misc.**shiftCoord**(*x*, *T*, *I*)

Map x in interval T to interval I.

Use an affine transformation to shift points x from the interval $T = [t_0, t_1]$ to the interval $I = [a, b]$.

Parameters

- **x** (*array_like*) – Points in interval T.
- **T** (*array_like*) – Original interval.
- **I** (*array_like*) – Target interval.

Returns
Shifted values for *x*.

Return type
array_like

pythia.misc.**str2iter**(*string*, *iterType=<class 'list'>*, *dataType=<class 'int'>*)

Cast *str(iterable)* to *iterType* of *dataType*.

Cast a string of lists, tuples, etc to the specified iterable and data type, i.e., for *iterType=tuple* and *dataType=float* cast `str([1,2,3]) -> (1.0, 2.0, 3.0)`.

Parameters

- **string** (*str*) – String representation of iterable.
- **iterType** (*iterable, default=list*) – Iterable type the string is converted to.
- **dataType** (*type, default=int*) – Data type of entries of iterable, e.g. *int* or *float*.

pythia.misc.**wlsSamplingBound**(*m*, *c=4*)

Compute the weighted Least-Squares sampling bound.

The number of samples n is chosen such that

$$\frac{n}{\log(n)} \geq cm,$$

where m is the dimension of the Gramian matrix (number of PC expansion terms) and c is an arbitrary constant. In Cohen & Migliorati 2017 the authors observed that the coice $c = 4$ yields a well conditioned Gramian with high probability.

Parameters

- **m** (*int*) – Dimension of Gramian matrix.
- **c** (*float, default=4*) – Scaling constant.

Returns
n – Number of required wLS samples.

Return type
int

1.1.10 pythia.parameter module

PyThia classes containing Parameter information.

class pythia.parameter.**Parameter**(*name*, *value=None*)

Bases: `object`

Basic parameter with fixed value.

Parameters

- **name** (*str*) – Name identifier of the parameter.
- **value** (*float, optional*) – Parameter value.

property mode

Type (mode) of parameter.

property name

Parameter name.

property value

Parameter value.

class pythia.parameter.**RandomParameter**(*name*, *domain*, *distribution*, *value=None*, *index=None*)

Bases: *Parameter*

Subclass of Parameter used for stochasic parameters.

Parameters

- **name** (*str*) – Parameter name.
- **domain** (*array_like*) – Supported domain of the parameter distribution.
- **distribution** (*str*) – Distribution identifier of the parameter.
- **value** (*float or array_like, optional*) – Value (expected) of the parameter. Only used to store information never required for computations.
- **index** (*int, optional*) – Enumeration index of the parameter. Set automatically if *None*.

property alpha

Parameter required for Beta or Gamma distribution.

property beta

Parameter required for Beta distribution.

property distribution

Distribution identifier of the parameter.

property domain

Supported domain of the parameter.

property index

Index of the parameter.

property mean

Mean of parameter distribution.

property variance

Variance of parameter distribution.

class pythia.parameter.**SteadyParameter**(*name*, *value*, *index=None*)

Bases: *Parameter*

Subclass of Parameter used for deterministic parameter variations.

Parameter representing part of the spatial/physical domain of the problem.

Parameters

- **name** (*str*) – Parameter name.
- **value** (*array_like*) – Values the parameter is iterated over.
- **index** (*int, optional*) – Enumeration index of the parameter. Set automatically if *None*.

property index

Index of the parameter.

property name

Parameter name.

property value

Parameter value.

1.1.11 pythia.sampler module

Sampler classes for generating in random samples and PDF evaluations.

class pythia.sampler.**BetaSampler**(*domain*, *alpha*, *beta*)

Bases: *Sampler*

Sampler for univariate Beta distributed samples on given domain.

Parameters

- **domain** (*array_like*) – Supported domain of distribution.
- **alpha** (*float*) – Parameter for Beta distribution.
- **beta** (*float*) – Parameter for Beta distribution.

property alpha

Parameter α of Gamma distribution.

property beta

Parameter β of Gamma distribution.

density(x)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in x.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

grad_x_log_pdf(x)

Evaluate gradient of log-PDF.

Note: Not yet implemented.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of gradient (vector valued) of log-PDF evaluated in *x*.

Return type
array_like

hess_x_log_pdf(*x*)

Evaluate Hessian of log-PDF.

Note: Not yet implemented.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of Hessian (matrix valued) of log-PDF evaluated in *x*.

Return type
array_like

log_pdf(*x*)

Evaluate log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of log-PDF evaluated in *x*.

Return type
array_like

property mass

Mass of the PDF.

property maximum

Maximum value of the PDF.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

class pythia.sampler.**ConstraintSampler**(*sampler*, *constraints*)

Bases: *Sampler*

Sampler that applies algebraic constraints when sampling.

Parameters

- **sampler** (*Sampler*) – Sampler to sample from.
- **constraints** (*list*) – List of functions that return True if sample point satisfies the constraint.

Notes

The constaints may lead to a non-normalized density function.

property constraints
Specified constrains.

density(*x*)
Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension
Dimension of the parameters.

property domain
Parameter domain of support.

property mass
Mass of the PDF.

property maximum
Maximum value of the PDF.

pdf(*x*)
Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

class pythia.sampler.**GammaSampler**(*domain*, *alpha*, *beta*)

Bases: *Sampler*

Sampler for univariate Gamma distributed samples on given domain.

Parameters

- **domain** (*array_like*) – Supported domain of distribution.
- **alpha** (*float*) – Parameter for Gamma distribution.
- **beta** (*float*) – Parameter for Gamma distribution.

property alpha

Parameter α of Gamma distribution.

property beta

Parameter β of Gamma distribution.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

grad_x_log_pdf(*x*)

Evaluate gradient of log-PDF.

Note: Not yet implemented.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of gradient (vector valued) of log-PDF evaluated in *x*.

Return type
array_like

`hess_x_log_pdf`(*x*)

Evaluate Hessian of log-PDF.

Note: Not yet implemented.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of Hessian (matrix valued) of log-PDF evaluated in *x*.

Return type
array_like

`log_pdf`(*x*)

Evaluate log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of log-PDF evaluated in *x*.

Return type
array_like

`property mass`

Mass of the PDF.

`property maximum`

Maximum value of the PDF.

`pdf`(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

`sample`(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

class pythia.sampler.**NormalSampler**(*mean*, *var*)

Bases: *Sampler*

Sampler for univariate normally distributed samples.

Parameters

- **mean** (*float*) – Mean of the Gaussian distribution.
- **var** (*float*) – Variance of the Gaussian distribution.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

grad_x_log_pdf(*x*)

Evaluate gradient of log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of gradient (vector valued) of log-PDF evaluated in *x*.

Return type
array_like

hess_x_log_pdf(*x*)

Evaluate Hessian of log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of Hessian (matrix valued) of log-PDF evaluated in *x*.

Return type
array_like

log_pdf(*x*)

Evaluate log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of log-PDF evaluated in *x*.

Return type
array_like

property mass

Mass of the PDF.

property maximum

Maximum value of the PDF.

property mean

Mean of the distribution.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

property var

Variance of the distribution.

class pythia.sampler.**ParameterSampler**(*params*)

Bases: *Sampler*

Product sampler of given parameters.

Parameters
params (list of *pythia.parameter.RandomParameter*) – List or dictionary containing information of parameters.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

property mass

Mass of the PDF.

Note: The PDF is assumed to be normalized. This is not computed here.

property maximum

Maximum value of the PDF.

property parameter

List of parameters.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

weight(*x*)

Weights for the parameter product PDF sampler.

class pythia.sampler.**ProductSampler**(*samplerList*)

Bases: *Sampler*

Tensor sampler for independent parameters.

Sampler for cartesian product samples of a list of (independent) univariate samplers.

Parameters
samplerList (list of *pythia.sampler.Sampler*) – List of (univariate) Sampler objects.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

`property dimension`

Dimension of the parameters.

`property domain`

Parameter domain of support.

`grad_x_log_pdf`(*x*)

Evaluate gradient of log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of gradient (vector valued) of log-PDF evaluated in *x*.

Return type
array_like

`hess_x_log_pdf`(*x*)

Evaluate Hessian of log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of Hessian (matrix valued) of log-PDF evaluated in *x*.

Return type
array_like

`log_pdf`(*x*)

Evaluate log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of log-PDF evaluated in *x*.

Return type
array_like

`property mass`

Mass of the PDF.

`property maximum`

Maximum value of the PDF.

`pdf`(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

`sample`(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

`class pythia.sampler.RejectionSampler`(*domain*, *pdf*, *trialSampler*, *maximum=None*)

Bases: *Sampler*

Rejection sampler for sampling arbitrary n-dimensional densities.

Parameters

- **domain** (*array_like*) – Domain of definition of density
- **pdf** (*function*) – Target density to generate samples from.
- **trialSampler** (*pythia.sampler.Sampler*) – Sampler for trial distribution, should generate samples fast.
- **maximum** (*float, optional*) – Threshold parameter with `density <= maximum * sampler.pdf`

`density`(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

`property dimension`

Dimension of the parameters.

`property domain`

Parameter domain of support.

`property mass`

Mass of the PDF.

Note: The PDF is assumed to be normalized. This is not computed here.

property maximum

Maximum value of the PDF.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

class pythia.sampler.**Sampler**

Bases: object

Base class for all continuous samplers.

property dimension

Dimension of the ambient space.

abstract property domain

Supported domain in ambient space.

The domain of a sampler is a list of pairs representing the n-orthotope (hyperrectangle) where it is defined.

abstract property mass

Mass of the sampler distribution.

The integral of the sampler distribution over the domain of definition. If the density is normalised this value should be one.

abstract pdf(*point*)

Density of the samplers distribution.

Computes the density of the samplers underlying distribution at the given *point*.

Parameters
point (*array_like of shape (..., D)*) – List of points or single point. *D* is the objects dimension.

Returns
out – Density at the point.

Return type
ndarray of shape (..., 1)

abstract sample(**shape*)

Random values in a given shape.

Create an array of the given shape and populate it with random samples from the samplers distribution.

Parameters
shape (*array_like, optional*) – The dimensions of the returned array, should all be positive. If no argument is given a single Python float is returned.

Returns
out – Random values of specified shape.

Return type
array_like

class pythia.sampler.**UniformSampler**(*domain*)

Bases: *Sampler*

Sampler for univariate uniformly distributed samples on given domain.

Parameters
domain (*array_like*) – Interval of support of distribution.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

grad_x_log_pdf(*x*)

Evaluate gradient of log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of gradient (vector valued) of log-PDF evaluated in *x*.

Return type
array_like

hess_x_log_pdf(*x*)

Evaluate Hessian of log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of Hessian (matrix valued) of log-PDF evaluated in *x*.

Return type
array_like

`log_pdf`(*x*)

Evaluate log-PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of log-PDF evaluated in *x*.

Return type
array_like

`property mass`

Mass of the PDF.

`property maximum`

Maximum value of the PDF.

`pdf`(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

`sample`(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

`class` pythia.sampler.`WLSConditionalSampler`(*params*, *basis*, *alphas*, *condition=<Mock name='mock.array()' id='140218929822544'>*)

Bases: *Sampler*

WLS univariate conditional Sampler as described by Cohen & Migliorati.

Parameters

- **params** (*dict*) – Dictionary of pythia.Parameters.Parameter.
- **basis** (*dict*) – Dictionary of basis functions for all parameters in params.
- **alphas** (*list*) – List of tuples representing multiindices.
- **condition** (*array_like*) – Array of condition for sampling.
- **note:** (..) – This is implemented very inefficiently.:

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

property mass

Mass of the PDF.

property maximum

Maximum value of the PDF.

property parameter

Parameter list.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property position

Parameter index.

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

property trialSampler

Trial sampler used for the rejection sampling.

class pythia.sampler.**WLSRejectionSampler**(*params*, *basis*, *tsa*)

Bases: *Sampler*

WLS rejection sampler.

Sampler for weighted Least-Squares sampling as described by Cohen and Migliorati. Uses rejection sampling for the weighted Least-Squares density.

Parameters

- **params** (list of *pythia.parameter.RandomParameter*) – List of parameters.
- **basis** (*dict*) – Dictionary of basis functions.
- **tsa** (*bool*) – Trial sampler adaptation. If True, a trial sampler is chosen on the distributions of parameters, if false a uniform trial sampler is used.

property basis

Employed basis functions.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

property mass

Mass of the PDF.

property maximum

Maximum value of the PDF.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

property trialSampler

Trial sampler used for the rejection sampling.

weight(*x*)

Weights for the PDF.

Parameters
x (*array_like*) – Points the weight function is evaluated in.

Returns
w – weights of evaluation points *x*.

Return type
array_like

class pythia.sampler.**WLSSampler**(*params*, *basis*, *strategy='rejection'*, *paramBasis=None*, *alphas=None*, *tsa=True*)

Bases: *Sampler*

Weighted Least-Squares sampler as in Cohen & Migliorati 2017.

Parameters

- **trialSampler** (*pythia.sampler.Sampler*) – Trial sampler to draw samples from.
- **basis** (*dict*) – Dictionary of basis functions.
- **strategy** (*string, ('rejection', 'sequential')*) – Strategy for Sampling.
- **paramBasis** (*dict of dict of function, optional*) – Univariate basis functions for all parameter.
- **alphas** (*list of tuple, optional*) – List of Multiindices represented by tuples.
- **tsa** (*bool, default=False*) – Trial sampler adaptation. If True, a trial sampler is chosen on the distributions of parameters, if false a uniform trial sampler is used.

property basis

Employed basis functions.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

property mass

Mass of the PDF.

property maximum

Maximum value of the PDF.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

property trialSampler

Trial sampler used for the rejection sampling.

weight(*x*)

Weights for the PDF sampler.

class pythia.sampler.**WLSSequentialSampler**(*params*, *basis*, *paramBasis*, *alphas*)

Bases: *Sampler*

WLS sequential conditional sampler as in Cohen & Migliorati 2017.

Parameters

- **params** (*dict of* pythia.parameter.RandomParameter) –
- **basis** (*dict of function*) – Multivariate basis functions.
- **paramBasis** (*dict of dict of function*) – Univariate basis functions for all parameter.
- **alphas** (*list of tuple*) – List of Multiindices represented by tuples.

property alphas

List of Multiindices represented by tuples.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

`property dimension`

Dimension of the parameters.

`property domain`

Parameter domain of support.

`property mass`

Mass of the PDF.

`property maximum`

Maximum value of the PDF.

`property parameter`

Parameter list.

`pdf`(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

`sample`(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

`weight`(*x*)

Weights for the PDF.

Parameters
x (*array_like*) – Points the weight function is evaluated in.

Returns
w – weights of evaluation points *x*.

Return type
array_like

class pythia.sampler.**WLSTensorSampler**(*params*, *deg*, *tsa=True*)

Bases: *Sampler*

WLS sampler for tensor multivariate basis.

Sampler for weighted Least-Squares sampling as described by Cohen and Migliorati. Only full tensor space can be sampled. This allows for univariate weighted Least-Squares sampling in each component.

Parameters

- **params** (list of *pythia.parameter.RandomParameter*) – Parameter list.
- **deg** (*int*) – Polynomial degree of each component (same for all).
- **tsa** (*bool, default=True*) – Trial sampler adaptation. If True, a trial sampler is chosen on the distributions of parameters, if false a uniform trial sampler is used.

property degree

Polynomial degree for each component.

density(*x*)

Evaluate PDF.

Deprecated since version 2.0.0: *density* will be removed in PyThia 3.0.0, use *pdf* instead.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

property dimension

Dimension of the parameters.

property domain

Parameter domain of support.

property mass

Mass of the PDF.

Note: The PDF is assumed to be normalized. This is not computed here.

property maximum

Maximum value of the PDF.

pdf(*x*)

Evaluate PDF.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of PDF evaluated in *x*.

Return type
array_like

sample(**shape*)

Draw samples from distribution.

Parameters
shape (*array_like*) – Shape of the samples.

Returns
Random samples of specified shape.

Return type
array_like

univariateDensity(*x*)

Evaluate univariate PDFs.

Parameters
x (*array_like*) – Evaluation points.

Returns
Values of univariate PDFs evaluated in *x*.

Return type
array_like

univariateWeight(*x*)

Weights for the univariate components.

Parameters
x (*array_like*) – Points the weight function is evaluated in.

Returns
w – weights of evaluation points *x*.

Return type
array_like

weight(*x*)

Weights for the PDF.

Parameters
x (*array_like*) – Points the weight function is evaluated in.

Returns
w – weights of evaluation points *x*.

Return type
array_like

pythia.sampler.**assign_sampler**(*param*)

Assign a univariate sampler to the given parameter.

Parameters
param (pythia.parameter.RandomParameter) –

Returns
Univariate sampler.

Return type
pythia.sampler.Sampler

pythia.sampler.**get_maximum**(*f*, *domain*, *nPoints=1000*)

Compute essential maximum of function by point evaluations.

Parameters

- **f** (*function*) – Function to evaluate. Needs to map from n-dim space to 1-dim space.
- **domain** (*array_like*) – Domain to evaluate function on.
- **nPoints** (*int, default=1000*) – Number of function evaluations. Evaluations are done on a uniform grid in domain. Actual number of points may thus be a little greater.

Returns

Approximation of maximum of function *f*.

Return type

float

1.1.12 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

F

G

H

I

L

M

N

P

R

S

T

U

V

W

X