
QMCPy

Release 1.3

Sou-Cheng T. Choi **Fred J. Hickernell**
Michael McCourt **Jagadeeswaran Rathinavel**
Aleksei Sorokin

Jul 03, 2022

CONTENTS

| | | |
|----------|--|----------|
| 1 | About Our QMC Software Community | 1 |
| 2 | License | 3 |
| 3 | QMCPy Documentation | 5 |
| 3.1 | Discrete Distribution Class | 5 |
| 3.1.1 | Abstract Discrete Distribution Class | 6 |
| 3.1.2 | Digital Net Base 2 | 6 |
| 3.1.3 | Lattice | 6 |
| 3.1.4 | Halton | 8 |
| 3.1.5 | IID Standard Uniform | 8 |
| 3.2 | True Measure Class | 9 |
| 3.2.1 | Abstract Measure Class | 9 |
| 3.2.2 | Uniform | 9 |
| 3.2.3 | Gaussian | 9 |
| 3.2.4 | Brownian Motion | 9 |
| 3.2.5 | Lebesgue | 9 |
| 3.2.6 | Continuous Bernoulli | 9 |
| 3.2.7 | Johnson’s SU | 9 |
| 3.2.8 | Kumaraswamy | 9 |
| 3.2.9 | SciPy Wrapper | 9 |
| 3.3 | Integrand Class | 9 |
| 3.3.1 | Abstract Integrand Class | 10 |
| 3.3.2 | Custom Function | 10 |
| 3.3.3 | Keister Function | 10 |
| 3.3.4 | Box Integral | 10 |
| 3.3.5 | European Option | 10 |
| 3.3.6 | Asian Option | 10 |
| 3.3.7 | Multilevel Call Options with Milstein Discretization | 10 |
| 3.3.8 | Linear Function | 10 |
| 3.3.9 | Sobol’ Indices | 10 |
| 3.4 | Stopping Criterion Algorithms | 10 |
| 3.4.1 | Abstract Stopping Criterion Class | 11 |
| 3.4.2 | Guaranteed Digital Net Cubature (QMC) | 11 |
| 3.4.3 | Guaranteed Lattice Cubature (QMC) | 11 |
| 3.4.4 | Bayesian Lattice Cubature (QMC) | 11 |
| 3.4.5 | Bayesian Digital Net Cubature (QMC) | 11 |
| 3.4.6 | CLT QMC Cubature (with Replications) | 11 |
| 3.4.7 | Guaranteed MC Cubature | 11 |
| 3.4.8 | CLT MC Cubature | 11 |

| | | |
|----------|--|-----------|
| 3.4.9 | Continuation Multilevel QMC Cubature | 11 |
| 3.4.10 | Multilevel QMC Cubature | 11 |
| 3.4.11 | Continuation Multilevel MC Cubature | 11 |
| 3.4.12 | Multilevel MC Cubature | 11 |
| 3.5 | Utilities | 11 |
| 4 | Demos | 13 |
| 5 | Indices and tables | 15 |
| 6 | Sponsors | 17 |
| 6.1 | Illinois Tech | 17 |
| 6.2 | Kamakura Corporation | 17 |
| 6.3 | SigOpt | 17 |
| | Python Module Index | 19 |
| | Index | 21 |

ABOUT OUR QMC SOFTWARE COMMUNITY

LICENSE

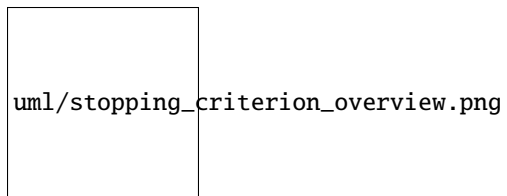
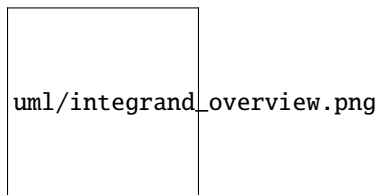
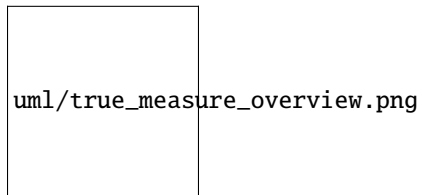
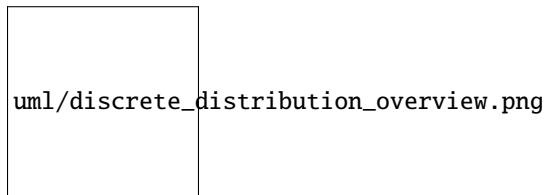
Copyright [2021] [Illinois Institute of Technology]

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

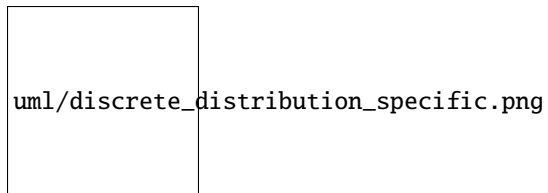
<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

QMCPY DOCUMENTATION



3.1 Discrete Distribution Class



3.1.1 Abstract Discrete Distribution Class

3.1.2 Digital Net Base 2

3.1.3 Lattice

```
class qmcpy.discrete_distribution.lattice.lattice.Lattice(dimension=1, randomize=True,
order='natural', seed=None, generating_vector='lattice_vec.3600.20.npy',
d_max=None, m_max=None)
```

Quasi-Random Lattice nets in base 2.

```
>>> l = Lattice(2, seed=7)
>>> l.gen_samples(4)
array([[0.04386058, 0.58727432],
       [0.54386058, 0.08727432],
       [0.29386058, 0.33727432],
       [0.79386058, 0.83727432]])
>>> l.gen_samples(1)
array([[0.04386058, 0.58727432]])
>>> l
Lattice (DiscreteDistribution Object)
  d          2^(1)
  dvec       [0 1]
  randomize  1
  order      natural
  entropy    7
  spawn_key  ()
>>> Lattice(dimension=2, randomize=False, order='natural').gen_samples(4, warn=False)
array([[0. , 0. ],
       [0.5 , 0.5 ],
       [0.25, 0.75],
       [0.75, 0.25]])
>>> Lattice(dimension=2, randomize=False, order='linear').gen_samples(4, warn=False)
array([[0. , 0. ],
       [0.25, 0.75],
       [0.5 , 0.5 ],
       [0.75, 0.25]])
>>> Lattice(dimension=2, randomize=False, order='mps').gen_samples(4, warn=False)
array([[0. , 0. ],
       [0.5 , 0.5 ],
       [0.25, 0.75],
       [0.75, 0.25]])
```

References

- [1] Sou-Cheng T. Choi, Yuhan Ding, Fred J. Hickernell, Lan Jiang, Lluís Antoni Jimenez Rugama, Da Li, Jagadeeswaran Rathinavel, Xin Tong, Kan Zhang, Yizhi Zhang, and Xuan Zhou, GAIL: Guaranteed Automatic Integration Library (Version 2.3) [MATLAB Software], 2019. Available from http://gailgithub.github.io/GAIL_Dev/
- [2] F.Y. Kuo & D. Nuyens. Application of quasi-Monte Carlo methods to elliptic PDEs with random diffusion coefficients - a survey of analysis and implementation, Foundations of Computational Mathematics, 16(6):1631-1696, 2016. springer link: <https://link.springer.com/article/10.1007/s10208-016-9329-5> arxiv link: <https://arxiv.org/abs/1606.06613>
- [3] D. Nuyens, *The Magic Point Shop of QMC point generators and generating vectors*. MATLAB and Python software, 2018. Available from <https://people.cs.kuleuven.be/~dirk.nuyens/>
- [4] Constructing embedded lattice rules for multivariate integration R Cools, FY Kuo, D Nuyens - SIAM J. Sci. Comput., 28(6), 2162-2188.
- [5] L'Ecuyer, Pierre & Munger, David. (2015). LatticeBuilder: A General Software Tool for Constructing Rank-1 Lattice Rules. ACM Transactions on Mathematical Software. 42. 10.1145/2754929.

```
__init__(dimension=1, randomize=True, order='natural', seed=None,
         generating_vector='lattice_vec.3600.20.npy', d_max=None, m_max=None)
```

Parameters

- **dimension** (*int* or *ndarray*) – dimension of the generator. If an int is passed in, use sequence dimensions [0,...,dimensions-1]. If a ndarray is passed in, use these dimension indices in the sequence.
- **randomize** (*bool*) – If True, apply shift to generated samples. Note: Non-randomized lattice sequence includes the origin.
- **order** (*str*) – ‘linear’, ‘natural’, or ‘mps’ ordering.
- **seed** (*None* or *int* or *numpy.random.SeedSeq*) – seed the random number generator for reproducibility
- **generating_vector** (*ndarray* or *str*) – generating matrix or path to generating matrices. ndarray should have shape (d_max). a string generating_vector should be formatted like ‘lattice_vec.3600.20.npy’ where ‘name.d_max.m_max.npy’
- **d_max** (*int*) – maximum dimension
- **m_max** (*int*) – 2^{m_max} is the max number of supported samples

Note: d_max and m_max are required if generating_vector is a ndarray. If generating_vector is a string (path), d_max and m_max can be taken from the file name if None

```
gen_samples(n=None, n_min=0, n_max=8, warn=True, return_unrandomized=False)
```

Generate lattice samples

Parameters

- **n** (*int*) – if n is supplied, generate from n_min=0 to n_max=n samples. Otherwise use the n_min and n_max explicitly supplied as the following 2 arguments
- **n_min** (*int*) – Starting index of sequence.
- **n_max** (*int*) – Final index of sequence.

- **return_unrandomized** (*bool*) – return samples without randomization as 2nd return value. Will not be returned if `randomize=False`.

Returns

(`n_max-n_min`) x `d` (dimension) array of samples

Return type

ndarray

Note: Lattice generates in blocks from 2^m to $2^{(m+1)}$ so generating `n_min=3` to `n_max=9` requires necessarily produces samples from `n_min=2` to `n_max=16` and automatically subsets. May be inefficient for non-powers-of-2 samples sizes.

pdf(x)

pdf of a standard uniform

3.1.4 Halton

3.1.5 IID Standard Uniform

`class qmcpy.discrete_distribution.iid_std_uniform.IIDStdUniform(dimension=1, seed=None)`

A wrapper around NumPy's IID Standard Uniform generator `numpy.random.rand`.

```
>>> dd = IIDStdUniform(dimension=2, seed=7)
>>> dd.gen_samples(4)
array([[0.04386058, 0.58727432],
       [0.3691824 , 0.65212985],
       [0.69669968, 0.10605352],
       [0.63025643, 0.13630282]])
>>> dd
IIDStdUniform (DiscreteDistribution Object)
  d          2^(1)
  entropy    7
  spawn_key  ()
```

`__init__(dimension=1, seed=None)`

Parameters

- **dimension** (*int*) – dimension of samples
- **seed** (*None or int or numpy.random.SeedSeq*) – seed the random number generator for reproducibility

gen_samples(n)

Generate samples

Parameters

n (*int*) – Number of observations to generate

Returns

`n` x `self.d` array of samples

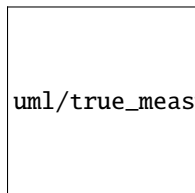
Return type

ndarray

$\text{pdf}(x)$

ABSTRACT METHOD to evaluate pdf of distribution the samples mimic at locations of x .

3.2 True Measure Class



uml/true_measure_specific.png

3.2.1 Abstract Measure Class

3.2.2 Uniform

3.2.3 Gaussian

3.2.4 Brownian Motion

3.2.5 Lebesgue

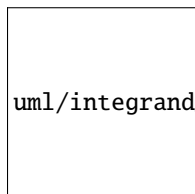
3.2.6 Continuous Bernoulli

3.2.7 Johnson's SU

3.2.8 Kumaraswamy

3.2.9 SciPy Wrapper

3.3 Integrand Class



uml/integrand_specific.png

3.3.1 Abstract Integrand Class

3.3.2 Custom Function

3.3.3 Keister Function

3.3.4 Box Integral

3.3.5 European Option

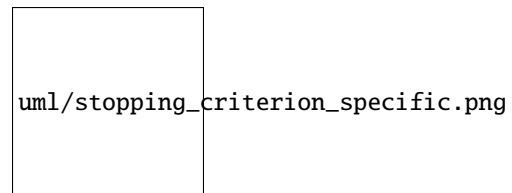
3.3.6 Asian Option

3.3.7 Multilevel Call Options with Milstein Discretization

3.3.8 Linear Function

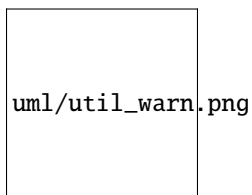
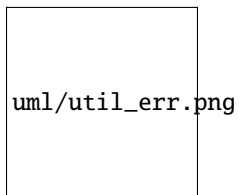
3.3.9 Sobol' Indices

3.4 Stopping Criterion Algorithms



- 3.4.1 Abstract Stopping Criterion Class
- 3.4.2 Guaranteed Digital Net Cubature (QMC)
- 3.4.3 Guaranteed Lattice Cubature (QMC)
- 3.4.4 Bayesian Lattice Cubature (QMC)
- 3.4.5 Bayesian Digital Net Cubature (QMC)
- 3.4.6 CLT QMC Cubature (with Replications)
- 3.4.7 Guaranteed MC Cubature
- 3.4.8 CLT MC Cubature
- 3.4.9 Continuation Multilevel QMC Cubature
- 3.4.10 Multilevel QMC Cubature
- 3.4.11 Continuation Multilevel MC Cubature
- 3.4.12 Multilevel MC Cubature

3.5 Utilities



`qmcpy.util.latnetbuilder_linker.latnetbuilder_linker`(*lnb_dir*='./', *out_dir*='./',
fout_prefix='lnb4qmcpy')

Parameters

- **lnb_dir** (*str*) – relative path to directory where *outputMachine.txt* is stored e.g. 'my_lnb/poly_lat'
- **out_dir** (*str*) – relative path to directory where output should be stored e.g. 'my_lnb/poly_lat_qmcpy/'
- **fout_prefix** (*str*) – start of output file name. e.g. 'my_poly_lat_vec'

Returns

path to file which can be passed into QMCPy's Lattice or Sobol' in order to use the linked latnetbuilder generating vector/matrix e.g. 'my_poly_lat_vec.10.16.npy'

Return type

str

Adapted from latnetbuilder parser:

https://github.com/umontreal-simul/latnetbuilder/blob/master/python-wrapper/latnetbuilder/parse_output.py#L74

**CHAPTER
FOUR**

DEMOS

INDICES AND TABLES

- genindex
- modindex
- search

SPONSORS

6.1 Illinois Tech

6.2 Kamakura Corporation

6.3 SigOpt

PYTHON MODULE INDEX

q

`qmcpy.discrete_distribution.iid_std_uniform`, 8
`qmcpy.discrete_distribution.lattice.lattice`, 6
`qmcpy.util.latnetbuilder_linker`, 11

Symbols

`__init__()` (*qmcpy.discrete_distribution.iid_std_uniform.IIDStdUniform method*), 8

`__init__()` (*qmcpy.discrete_distribution.lattice.lattice.Lattice method*), 7

G

`gen_samples()` (*qmcpy.discrete_distribution.iid_std_uniform.IIDStdUniform method*), 8

`gen_samples()` (*qmcpy.discrete_distribution.lattice.lattice.Lattice method*), 7

I

`IIDStdUniform` (*class in qmcpy.discrete_distribution.iid_std_uniform*), 8

L

`latnetbuilder_linker()` (*in module qmcpy.util.latnetbuilder_linker*), 11

`Lattice` (*class in qmcpy.discrete_distribution.lattice.lattice*), 6

M

module

qmcpy.discrete_distribution.iid_std_uniform, 8

qmcpy.discrete_distribution.lattice.lattice, 6

qmcpy.util.latnetbuilder_linker, 11

P

`pdf()` (*qmcpy.discrete_distribution.iid_std_uniform.IIDStdUniform method*), 8

`pdf()` (*qmcpy.discrete_distribution.lattice.lattice.Lattice method*), 8

Q

`qmcpy.discrete_distribution.iid_std_uniform`
 module, 8

`qmcpy.discrete_distribution.lattice.lattice`
 module, 6

`qmcpy.util.latnetbuilder_linker`
 module, 11