QMCPy: A Quasi-Monte Carlo (QMC) Software in Python 3

Aleksei Sorokin[1], Sou-Cheng Choi[1,2],
Fred Hickernell[1], Michael McCourt[3], Jagadeeswaran Rathinavel[1]

[1]Illinois Institute of Technology, [2]Kamakura Corporation, [3]SigOpt

Kamakura Corporation SIGOPT

The QMC Problem

Original Form

$$\mu = \int_{\mathcal{T}} g(\boldsymbol{t}) \lambda(\mathrm{d}\boldsymbol{t})$$

$g : \mathcal{T} \to \mathbb{R}$ = original integrand
λ = true measure

Convenient Form

$$\mu = \int_{\mathcal{T}} g(\boldsymbol{t}) \lambda(\mathrm{d}\boldsymbol{t}) = \int_{\mathcal{X}} f(\boldsymbol{x}) \nu(\mathrm{d}\boldsymbol{x})$$

ν = well defined probability measure
$T : \mathcal{X} \to \mathcal{T}$ = change of variables
$f : \mathcal{X} \to \mathbb{R}$ = integrand after change of variables

(Quasi-)Monte Carlo Approximation

$$\hat{\mu}_n = a_n \sum_{i=1}^{n} f(\boldsymbol{x}_i) w_i = \int_{\mathcal{X}} f(\boldsymbol{x})\, \hat{\nu}(\mathrm{d}\boldsymbol{x})$$

$$\nu \approx \hat{\nu}_n = a_n \sum_{i=1}^{n} w_i \delta_{\hat{\boldsymbol{x}}_i}(\cdot)$$

= discrete distribution

QMCPy Sources

- Package Distribution with PyPI
  `pypi.org/project/qmcpy`
- Open source code on GitHub
  `github.com/QMCSoftware/QMCSoftware`
- Documentation on Read the Docs
  `qmcpy.readthedocs.io/en/latest`
- Blogs posts on WordPress site
  `qmcpy.wordpress.com`
- Updates from QMC Software Google Group
  `qmc-software@googlegroups.com`

Installation

To install QMCPy with Python 3 run the command

```
pip install qmcpy
```

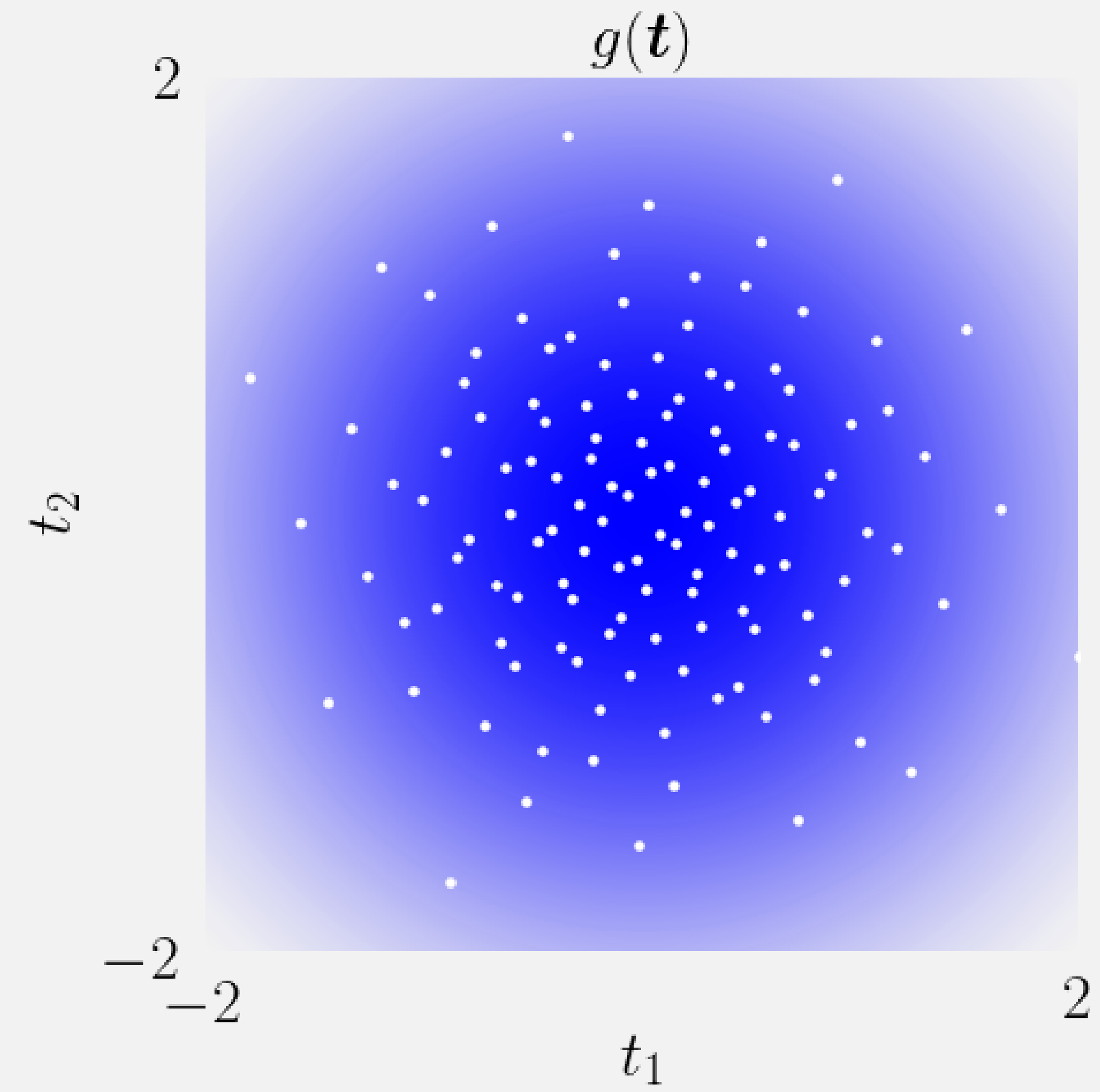
Keister Example

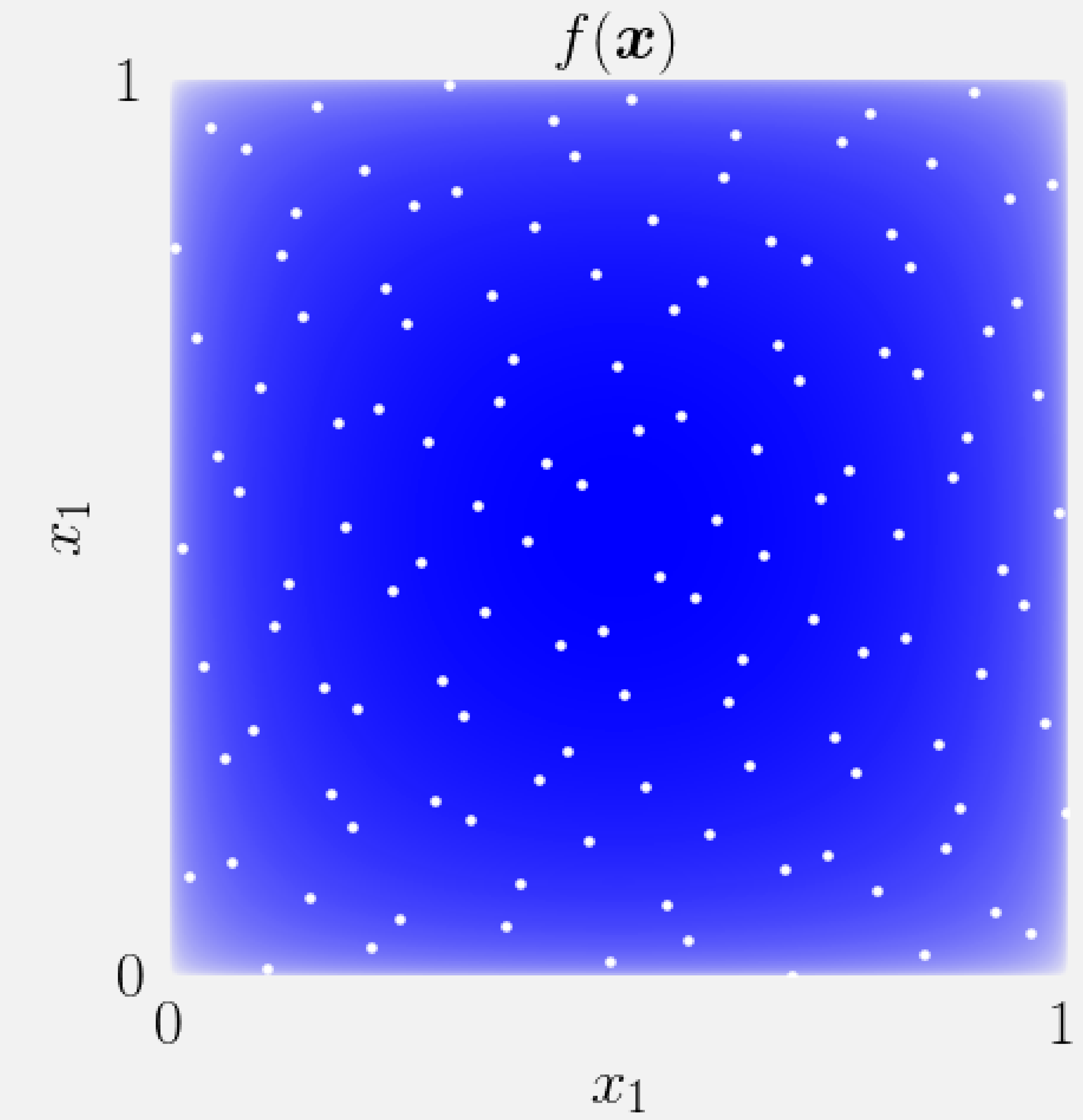
Original integrand [1]: $g(\boldsymbol{t}) = \pi^{d/2} \cos(||\boldsymbol{t}||)$
True measure: $\lambda \sim \mathcal{N}(\boldsymbol{0}, \mathsf{I}/2)$
Discrete distribution: $\hat{\nu} \sim$ Lattice

$$\therefore \quad \nu \sim \mathcal{U}(\boldsymbol{0},\boldsymbol{1}), \qquad T(\boldsymbol{x}) = \Phi^{-1}(\boldsymbol{x})/2$$

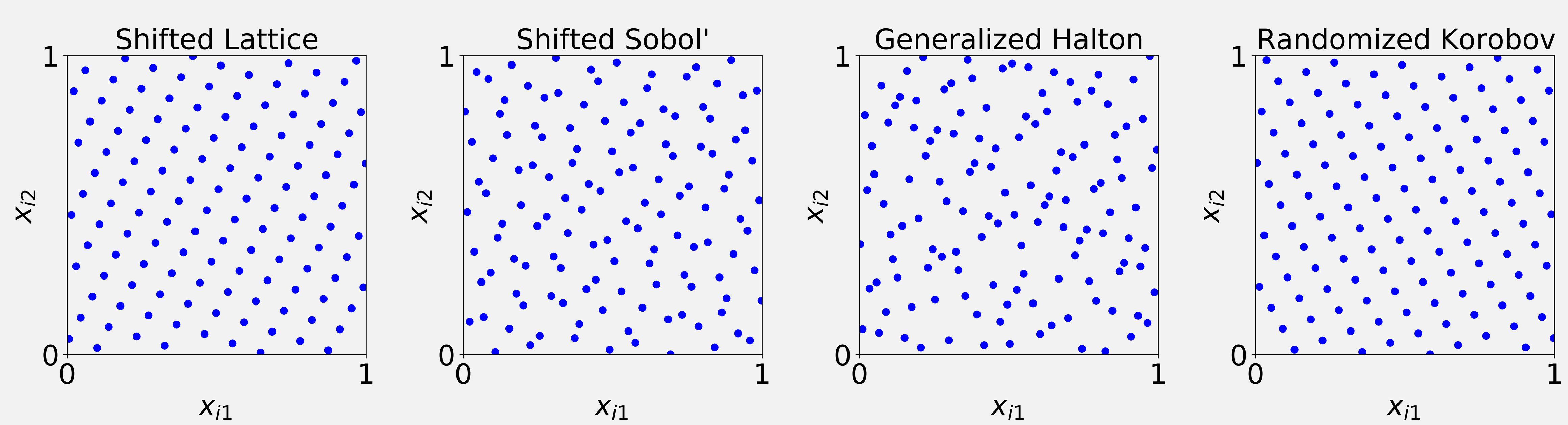
$$\mu = \int_{\mathbb{R}^d} g(\boldsymbol{t})\, \pi^{-d/2} \exp(-||\boldsymbol{t}||^2)\, \mathrm{d}\boldsymbol{t} = \int_{[0,1]^d} \pi^{d/2} \cos\left(||T(\boldsymbol{x})||\right)\, \mathrm{d}\boldsymbol{x}$$

```python
import qmcpy as qp
from numpy import *
d = 2
s = qp.Sobol(d)
g = qp.Gaussian(s, covariance=1/2)
k = qp.CustomFun(g, lambda x:
pi**(d/2)*cos(linalg.norm(x,axis=1)))
cs = qp.CubQMCSobolG(k, abs_tol=1e-3)
solution,data = cs.integrate()
```





Low Discrepancy Discrete Distributions




```
>>> qp.Lattice(dimension=2,randomize=True).gen_samples(n=2**7)
```

Contributing Projects

- Guaranteed Automatic Integration Library [2]
- Quasi-Random Number Generators [3]
- Giles' Et al. Multilevel MC [4] and QMC [5]
- Magic Point Shop [6]
- Owen's Halton Generator [7]
- Lattice generating vector from LatticeBuilder [8]

References

1. Keister, B. D. Multidimensional Quadrature Algorithms.Computers in Physics10,119–122(1996).
2. S.-C. T. Choi, Y. Ding, F. J. Hickernell, L. Jiang, Ll. A. Jiménez Rugama, D. Li, R. Jagadeeswaran, X. Tong, K. Zhang, Y. Zhang, and X. Zhou, "GAIL: Guaranteed Automatic Integration Library (versions 1.0-2.3)," MATLAB software, 2013-2019.
3. Marius Hofert and Christiane Lemieux (2019). qrng: (Randomized) Quasi-Random Number Generators. R package version 0.0-7.
4. M.B. Giles. 'Improved multilevel Monte Carlo convergence using the Milstein scheme'. 343-358, in Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer, 2008.
5. M.B. Giles and B.J. Waterhouse. 'Multilevel quasi-Monte Carlo path simulation'. pp.165-181 in Advanced Financial Modelling, in Radon Series on Computational and Applied Mathematics, de Gruyter, 2009.
6. F. Y. Kuo and 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.
7. Owen, A. B. A randomized Halton algorithm in R2017. arXiv:1706.02808 [stat.CO]
8. 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.