QMCPy
Quasi-Monte Carlo Community Software
Python 3

Aleksei Sorokin
Sou-Cheng T. Choi, Fred J. Hickernell, Lynn Matar, Mike McCourt

Illinois Institute of Technology
Department of Applied Mathematics
Computational Math Seminar

November 15, 2019

Original & Convenient Forms

$$\mu = \int_T g(t)\,\lambda(\mathrm{d}t) = \int_X f(x)\rho(x)\,\mathrm{d}x = \int_X f(x)\,\nu(\mathrm{d}x)$$

$g : T \to \mathbb{R} =$ original integrand
$\lambda =$ original measure
$\phi : X \to T =$ change of variables
$f : X \to \mathbb{R} =$ integrand after change of variables
$\nu =$ well understood probability measure

(Quasi-)Monte Carlo Approximation

$$\hat{\mu}_n = a_n \sum_{i=1}^{n} f(x_i) w_i = \int_X f(x)\, \hat{\nu}(\mathrm{d}x)$$

$$\begin{aligned} \nu \approx \hat{\nu}_n &= a_n \sum_{i=1}^{n} w_i \delta_{x_i}(\cdot) \\ &= \text{discrete probability measure} \end{aligned}$$

How to choose nodes $\{x_i\}_{i=1}^n$ so that $|\mu - \hat{\mu}_n| < \epsilon$?
ϵ = user-given error tolerance

Abstract Classes

Integrand

- $g : T \to \mathbb{R}$ = original integrand
- $f : X \to \mathbb{R}$ = integrand after change of variables

True Measure

- λ = original measure
- $\phi : X \to T$ = change of variables

Discrete Distribution

- ν = well defined probability measure

Stopping Criterion

- Find n

Accumulate Data

- House integration data

Inputs and Outputs of the `integrate` Method

Integrand

- Keister Function, Asian Call Option

True Measure

- Uniform, Gaussian, Brownian Motion, Lebesgue

Discrete Distribution

- (iid): Standard Gaussian, Standard Uniform
- (lds): Lattice, Sobol

Stopping Criterion

- (iid): Based on Central Limit Theorem (CLT)
- (lds): Based on Repeated CLT (CLTRep)

Accumulate Data

- $\hat{\mu}, \hat{\sigma}^2$ for CLT, CLTRep

Integrand

Keister Function [Kei96]

$$y = g(x) = \pi^{d/2} \cos(||x||_2)$$

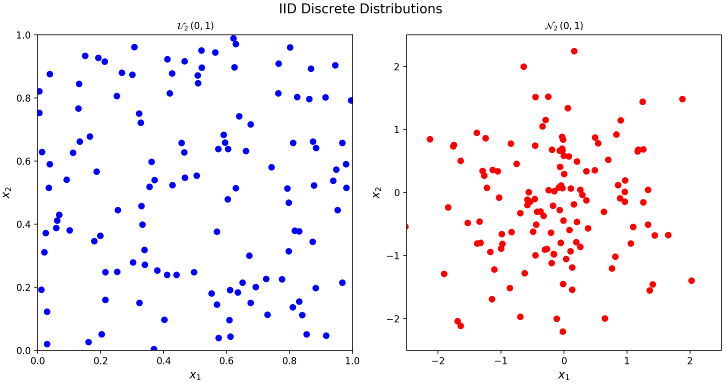
```python
class Keister(Integrand):
    def g(self, x):
        dimension = x.shape[1]
        normx = norm(x, 2, axis=1)
        y = pi ** (dimension / 2.0) * cos(normx)
        return y
integrand = Keister()
```

Equivalent construction

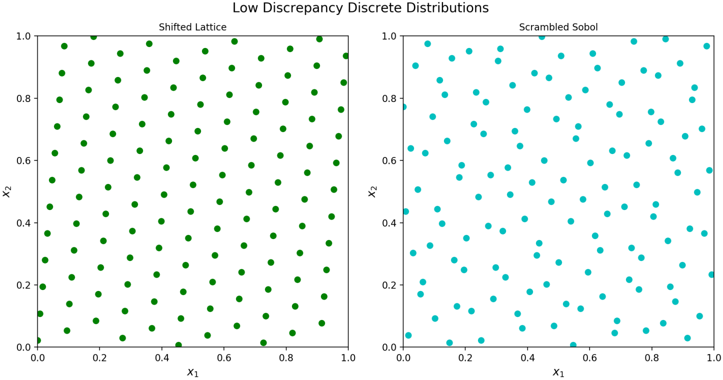
```python
integrand = QuickConstruct(\
    lambda x: pi**(x.shape[1]/2) * \
                cos(norm(x, 2, axis=1)))
```

IID

```python
# Generate X = [x1,x2] for left plot
dd = IIDStdUniform(rng_seed = 7)
X = dd.gen_dd_samples(1, 128, 2).squeeze()
```

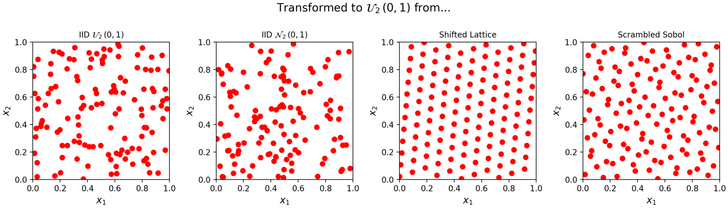


Low Discrepancy Sequence (lds)



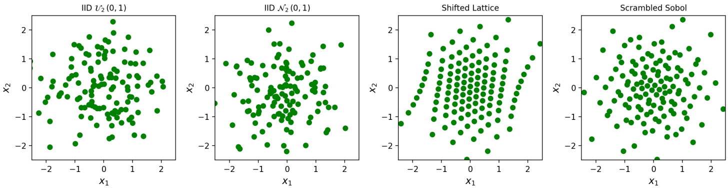
Uniform

```
# Generate X = [x1,x2] for right-most plot
tm = Uniform(dimension = 2)
dd = Sobol(rng_seed = 7)
tm.set_tm_gen(dd) # Initialize below method
X = tm.gen_tm_samples(r=1, n=128).squeeze()
```

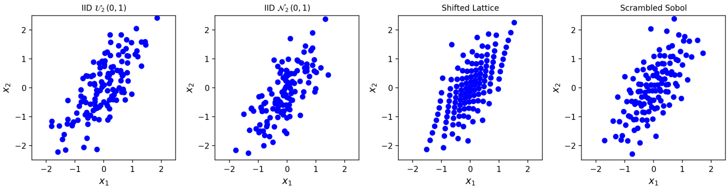


Gaussian

Transformed to $\mathcal{N}_2(0,1)$ from...

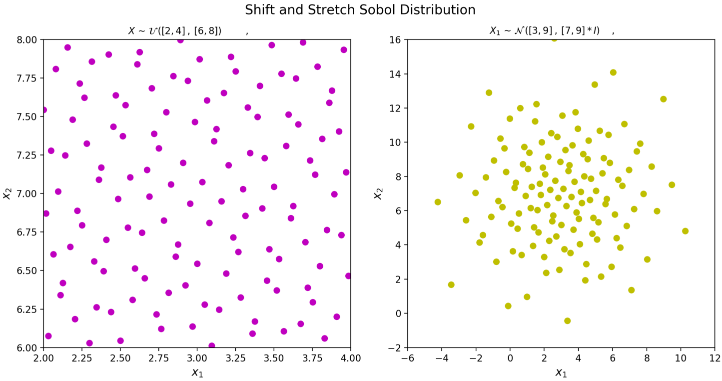


Transformed to Discretized BrownianMotion with time_vector = [.5 , 1] from...



Shift and Stretch

```
# Generate X = [x1,x2] for right plot
tm = Gaussian(dimension=[2], \
                mean=[[3,7]], variance=[[9,9]])
dd = Sobol(rng_seed = 7)
tm.set_tm_gen(dd) # Initialize below method
X = tm.gen_tm_samples(r=1, n=128).squeeze()
```



CLT Based Monte Carlo Algorithm for IID Nodes

1. Choose n_σ for pilot sample and compute $\hat{\sigma}^2_{n_\sigma}$
2. For a 99% confidence interval and inflation factor C, let:

$$n_\mu = \underset{n}{\text{argmin}}\left(\frac{2.58\, C\, \hat{\sigma}_{n_\sigma}}{\sqrt{n}} \leq \epsilon\right)$$

3. Compute $\hat{\mu}_{n_\mu}$ and $\hat{\epsilon} = \frac{2.58 C \hat{\sigma}_n}{\sqrt{n}}$ s.t.

$$\mathbb{P}[|\mu - \hat{\mu}_{n_\mu}| \leq \hat{\epsilon} \leq \epsilon] \geq 99\%$$

CLT Based Monte Carlo Algorithm for LDS Nodes

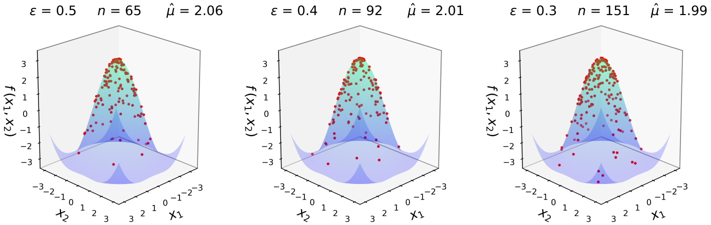
1. Choose $n = \frac{n_0}{2}$ and number or replications R
2. DO
   2.1 $n = 2n$
   2.2 Generate samples $\{X_j\}_{j=1}^R$ to compute $\{\hat{\mu}_{j,n}\}_{j=1}^R$
   2.3 Let $\hat{\sigma}_n = Std(\{\hat{\mu}_{j,n}\}_{j=1}^R)$

   WHILE $\hat{\sigma}_n \ > \ \epsilon$
3. Compute $\hat{\mu}_n = Mean(\{\hat{\mu}_{j,n}\}_{j=1}^R)$ and $\hat{\epsilon} = \frac{2.58\, C\, \hat{\sigma}_n}{\sqrt{n}}$ s.t.

$$\mathbb{P}[|\mu - \hat{\mu}_n| \leq \hat{\epsilon} \leq \epsilon] \geq 99\%$$

Keister Example

```
integrand = Keister()
dd = IIDStdGaussian(rng_seed=7)
tm = Gaussian(dimension=2, variance=1/2)
stop = CLT(dd, tm, abs_tol= ε, n_init=16)
sol, data = integrate(integrand, tm, dd, stop)
```



Keister Example Output

```
print(data)
"""
Solution: 2.0554
Keister (Integrand Object)
IIDStdGaussian (Discrete Distribution Object)
  mimics              StdGaussian
  rng_seed            7
Gaussian (True Measure Object)
  dimension           2
  mu                  0
  sigma               0.7071067811865476
CLT (Stopping Criterion Object)
  abs_tol             0.500
  rel_tol             0
  n_max               10000000000
  alpha               0.010
  inflate             1.200
MeanVarData (AccumData Object)
  n                   65
  n_total             81.0
  confid_int          [ 1.646  2.464]
  time_total          0.002              """
```

Multi-Level Asian Call Option Example [GS18]

```
tm = BrownianMotion(dimension = [4, 16, 64], \
        time_vector = \
            [
                arange(1/4, 5/4, 1/4),
                arange(1/16, 17/16, 1/16),
                arange(1/64, 65/64, 1/64)
            ])
integrand = \
    AsianCall(tm,
        volatility = .5, \
        start_price = 30, \
        strike_price = 25, \
        interest_rate = .01, \
        mean_type = 'arithmetic')
dd = IIDStdGaussian(rng_seed = 7)
stop = CLT(dd, tm, abs_tol=.05)
sol, data = integrate(integrand, tm, dd, stop)
```

Future Work

Attract collaborators

- i.e. Lattice, Sobol generators from Magic Point Shop [KN16]

Expand library of components & test cases

- Integrand, True Measure, Discrete Distribution, Stopping Criterion

Implement GAIL algorithms [CDH+19]

- meanMC_g, cubLattice_g, cubSobol_g [HCJ+18]

Project Links

[HCS19]

- GitHub (https://github.com/QMCSoftware/QMCSoftware.git)
- Documentation (https://qmcsoftware.github.io/QMCSoftware/index.html)
- Website (https://sites.google.com/hawk.iit.edu/qmc-software/home)

References I

Sou-Cheng T. Choi, Yuhan Ding, Fred J. Hickernell, Lan Jiang, Lluis 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]*, `http://gailgithub.github.io/GAIL_Dev/`, 2019.

Michael B Giles and Lukasz Szpruch, *Multilevel Monte Carlo methods for applications in finance*, High-Performance Computing in Finance, Chapman and Hall/CRC, 2018, pp. 197–247.

Fred J. Hickernell, Sou-Cheng T. Choi, Lan Jiang, Lluís Antoni, and Jiménez Rugama, *Monte Carlo Simulation, Automatic Stopping Criteria for*, Wiley StatRef: Statistics Reference Online (2018).

References II

- Fred J. Hickernell, Sou-Cheng T. Choi, and Aleksei Sorokin, *QMCPy: QMC Community Software*, https://github.com/QMCSoftware/QMCSoftware.git, 2019.
- B. D. Keister, *Multidimensional Quadrature Algorithms*, vol. 10, Computers in Physics, 1996.
- 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*, https://people.cs.kuleuven.be/~dirk.nuyens/qmc-generators/, 2016.