Community Supported Quasi-Monte Carlo (QMC) Software

Aleksei Sorokin (asorokin@hawk.iit.edu), Sou-Cheng Choi (schoi32@iit.edu), Fred Hickernell (hickernell@iit.edu)

Department of Applied Mathematics, Illinois Institute of Technology

Software Objectives

To provide QMC software [1] that is:

- comprised of free open source tools
- designed for continuous development
- easy to use for non-experts
- a recognized standard

The Integration Problem

Original Form

$$\mu = \int_T g(t)\,\lambda(\mathrm{d}t)$$

$g : T \to \mathbb{R}$ is original integrand
λ is original measure

Convenient Form

$$\mu = \int_X f(x)\rho(x)\,dx = \int_X f(x)\,\nu(\mathrm{d}x)$$

$f : X \to \mathbb{R}$ is integrand after change of variables
ν is some well defined probability measure
$\rho : X \to T$ is the probability density

(Quasi-)Monte Carlo Approximation

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

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

Main Object Classes

The routine `integrate` accepts instances of concrete subclasses of four abstract classes that represent the integrand, original measure, discrete distribution associated with the sampling points, and stopping criterion.

`integrate`

Given $\epsilon > 0$, find $\hat{\mu}_n$ such that $|\mu - \hat{\mu}_n| \le \epsilon$

Arguments

- Function instance
- Measure instance
- Discrete Distribution instance
- Stopping Criteria instance

Function

Specify and generate values $f(x_i)$

Concrete Classes

- Keister functions [2]
- Asian call option's payoff

Discrete Distribution, $\hat{\nu}$

Specify and generate $a_n \sum_{i=1}^{n} w_i\delta_{x_i}(\cdot)$

Concrete Classes

- IID
- QMC (Lattice & Sobol) [3, 4]

Stopping Criterion

Determine sample size, n, given ϵ

Concrete Classes

- Central limit theorem (IID)
- Mean Variance (Mesh)

Measure, λ and ν

Specify the original measure and convienient probability measure

Implemented Functions

- Standard uniform
- Standard Gaussian
- IID zero-mean Gaussian
- Brownian motion

Accumulate Data

Accumulate data required for the computation of the integral and the stopping criterion

Python Examples

Listing 1: Essential code for estimating the Keister integral in the first subplot of Figure 1.

```
d = 2
stopObj = CLTStopping(nInit=16,
                      absTol=.5)
measureObj = measure().\
    IIDZMeanGaussian(dimension=[d],
                     variance=[.5])
distribObj = IIDDistribution(
    trueD=measure().stdGaussian(
        dimension=[d]))
sol, out = integrate(KeisterFun(),
    measureObj, distribObj, stopObj)
```

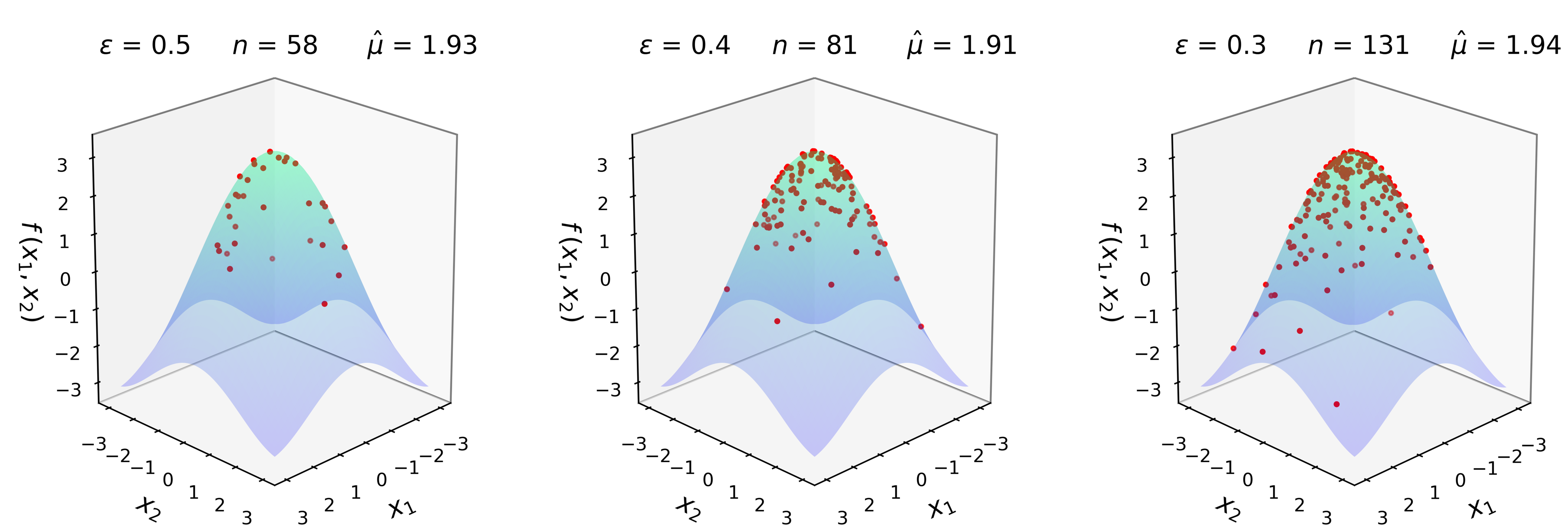

Figure 1: Reducing tolerance automatically results in more samples and better approximations of the Keister integral [2], $\int_{\mathbb{R}^d} \pi^{d/2} \cos(\|\boldsymbol{t}\|) \frac{\exp(-\|\boldsymbol{t}\|^2)}{\pi^{d/2}}\,\mathrm{d}\boldsymbol{t} = \int_{\mathbb{R}^d} \pi^{d/2} \cos(\|\boldsymbol{x}\|/\sqrt{2}) \frac{\exp(-\|\boldsymbol{x}\|^2/2)}{(2\pi)^{d/2}}\,\mathrm{d}\boldsymbol{x} \approx 1.80819$ with $d = 2$ and where Φ is the standard normal cumulative distribution function. This figure is reproducible by `Test_3D_Point_Distribution.py` [1].

Results

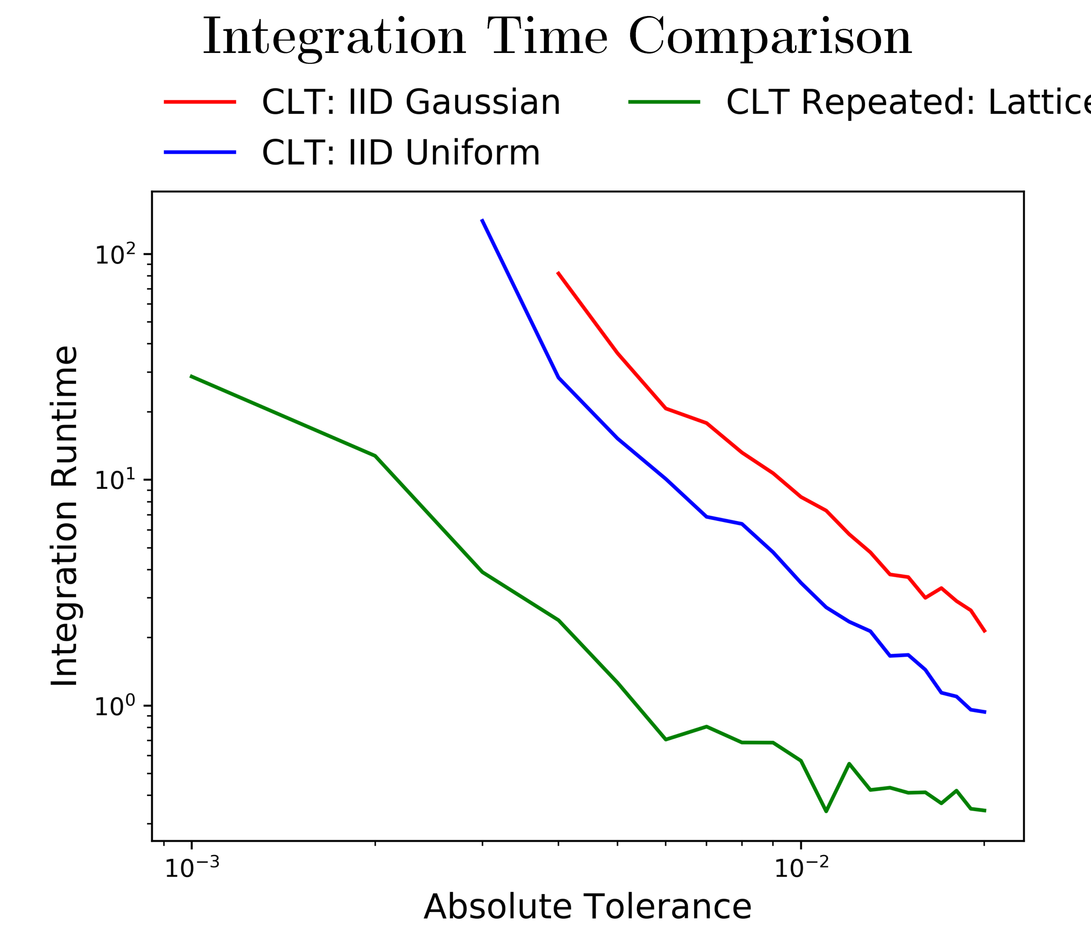

Figure 2: Multi-dimensional Asian call option integrated with respect to Brownian Motion. This figure can be easily reproduced by `Test_AbsTol_RunTime.py` [1].

Future Work

- Enhance tests, examples, and documentation
- Refine existing code. e.g. improve Sobol speed
- Bring in relevant algorithms from GAIL [5]
- Expand community of contributors

References

[1] F. J. Hickernell, S.-C. T. Choi, and A. Sorokin, "QMC Community Software." MATLAB and Python 3 software, 2019. Work in progress. https://github.com/QMCSoftware/QMCSoftware.

[2] B. D. Keister, "Multidimensional quadrature algorithms," *Computers in Physics*, vol. 10, no. 2, pp. 119–128, 1996.

[3] 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*, vol. 16, no. 6, pp. 1631–1696, 2016.

[4] D. Nuyens, "The magic point shop of QMC point generators and generating vectors." MATLAB and Python software, 2018. https://people.cs.kuleuven.be/ dirk.nuyens/qmc-generators.

[5] 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. http://gailgithub.github.io/GAIL_Dev/.