QMCPy: Quasi-Monte Carlo Software in Python

Aleksei G. Sorokin[1], Sou-Cheng T. Choi[1,2], Fred J. Hickernell[1], Michael J. McCourt[3], Jagadeeswaran Rathinavel[4]

[1]Illinois Institute of Technology, [2]Kamakura Corporation, [3]SigOpt, [4]Wi-Tronix

Thank you to NSF Grant DMS-1522687 and [3]SigOpt for supporting QMCPy

Numerical Integration Techniques

Error in d dimensions with n points:

Sparse Grids: $\mathcal{O}(n^{-1/d})$

Monte Carlo (MC): $\mathcal{O}(n^{-1/2})$

Quasi-Monte Carlo (QMC): $\mathcal{O}(n^{-1})$

MC & QMC avoid the curse of dimensionality. QMC is more efficient than MC.

Applications

Applied Statistics, Finance, Computer Graphics, AI, Computational Sciences, Engineering, Simulation, ...

The (Quasi-)Monte Carlo Problem

Integral Transform

$$\mu = \int_{\mathcal{T}} g(\boldsymbol{t})\lambda(\boldsymbol{t})\mathrm{d}\boldsymbol{t} = \int_{[0,1]^d} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$

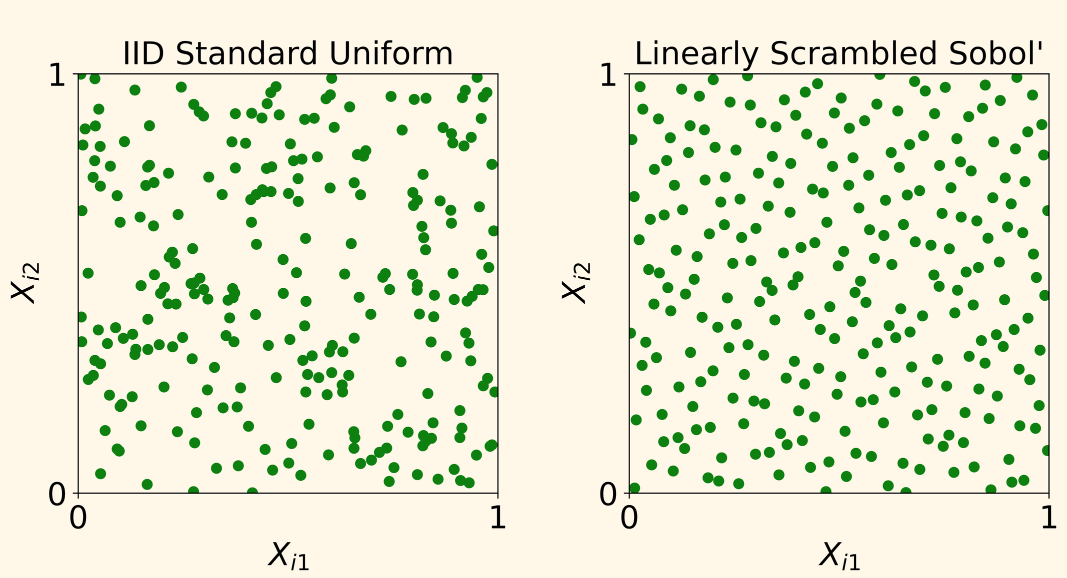
$g : \mathcal{T} \to \mathbb{R} =$ original integrand
$\lambda =$ true measure weight
$\boldsymbol{T} : [0,1]^d \to \mathcal{T} =$ change of variables
$f : [0,1]^d \to \mathbb{R} =$ integrand after change of variables

(Quasi-)Monte Carlo Approximation

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i) \approx \int_{[0,1]^d} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \mu$$

discrete distribution $= \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\} \sim \mathcal{U}[0,1]^d$

Monte Carlo vs Quasi-Monte Carlo



Choose discrete distribution $\{\boldsymbol{x}_i\}_{i=1}^n$ to be

IID (independent ...) - Monte Carlo (left)

LD (low discrepancy) - Quasi-Monte Carlo (right)

Pricing an Asian Call Option

Time Horizon $\tau = 1$ Interest Rate $r = 0$ Volatility $\sigma = 0.5$ Start Price $S_0 = 100$ Strike Price $K = 150$

True Measure $\boldsymbol{T} = (T_1, \dots, T_d) \sim \mathcal{N}(\boldsymbol{0}, \Sigma)$, $\quad \Sigma = (\tau/d)(\min(k,j))_{j,k=1}^d$

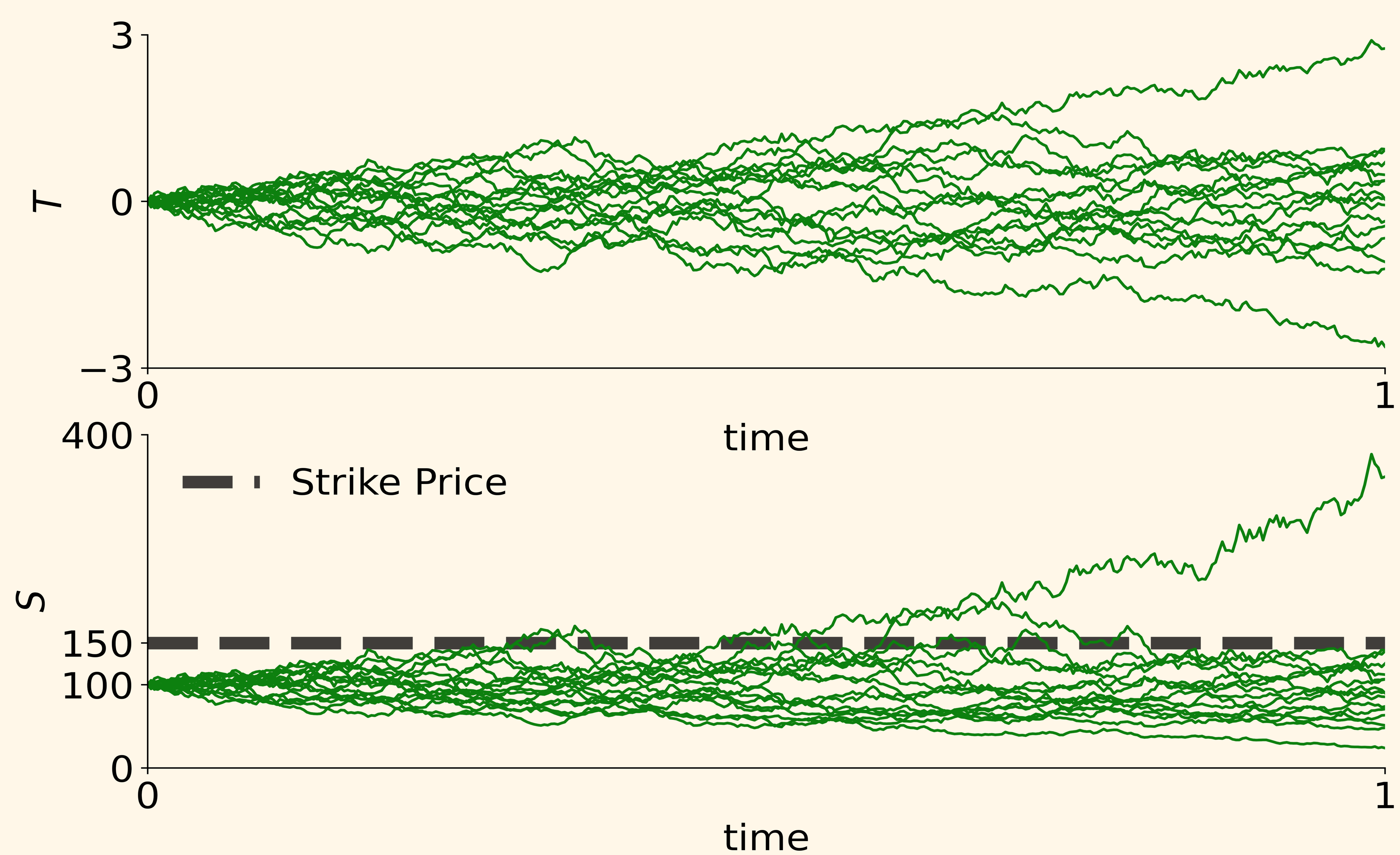
Asset Path $S_j(\boldsymbol{T}) = S_0 \exp((r - \sigma^2/2)\tau j/d + \sigma T_j)$, $\quad j = 1, \dots, d$

Payoff $\mathrm{Pay}(\boldsymbol{S}) = \max\left(\frac{1}{2d}\sum_{j=1}^{d}(S_{j-1} + S_j) - K, 0\right)$, $\quad \boldsymbol{S} = (S_0, \dots, S_d)$

Discounted Payoff $g(\boldsymbol{t}) = \mathrm{Pay}(\boldsymbol{S}(\boldsymbol{t}))\exp(-r\tau)$

$$\mu = \mathbb{E}(g(\boldsymbol{T})) = \int_{\mathbb{R}^d} g(\boldsymbol{t}) \underbrace{(2\pi)^{-d/2}\det(\Sigma)^{-1/2}\exp\left(-\boldsymbol{t}^T\Sigma^{-1}\boldsymbol{t}/2\right)}_{\mathcal{N}(\boldsymbol{0},\Sigma)\text{ density}} \mathrm{d}\boldsymbol{t}$$

```
>>> import qmcpy as qp
>>> sobol = qp.Sobol(dimension=365) # LD sequence, daily monitoring
>>> params = {"start_price":100, "strike_price":150, "call_put":"call"}
>>> aco = qp.AsianOption(sobol,**params) # defaults other params
>>> algorithm = qp.CubQMCSobolG(aco,abs_tol=1e-2) # tol of a penny
>>> solution,data = algorithm.integrate() # determine n to meet tol
>>> print("Approx discounted payoff $%.2f took %.2f seconds, %d samples."%
...  (solution,data.time_integrate,data.n_total)) # print(data) for details
Approx discounted payoff $1.47 took 0.81 seconds, 16384 samples.
```



Contributing Projects

- Guaranteed Automatic Integration Library [3]
- Quasi-Random Number Generators [4]
- P. Robbe's Multilevel MC and QMC [5]
- M. Giles' Multilevel MC and QMC [6,7]
- A. Owen's Halton Generator [8]
- LatNet Builder Generating Vectors [9]

References

1 S.-C. T. Choi, F. J. Hickernell, R. Jagadeeswaran, M. J. McCourt, A. G. Sorokin, "Quasi-Monte Carlo Software," Monte Carlo and Quasi-Monte Carlo Methods 2020.

2 S.-C. T. Choi, F. J. Hickernell, R. Jagadeeswaran, M. J. McCourt, A. G. Sorokin, "QMCPy: A Quasi-Monte Carlo Python Library," www.qmcpy.org.

3 S.-C. T. Choi, Y. Ding, F. J. Hickernell, L. Jiang, D. Li, R. Jagadeeswaran, L.-A. Jimenez Rugama, X. Tong, K. Zhang, Y. Zhang, and X. Zhou, "GAIL: Guaranteed Automatic Integration Library," Version 2.3.1, MATLAB Software, 2020.

4 M. Hofert and C. Lemieux, "qrng: (Randomized) Quasi-Random Number Generators," R package version 0.0-7, (2019).

5 P. Robbe. "Multilevel Uncertainty Quantification Methods for Robust Design of Industrial Applications," PhD thesis, KU Leuven, 2019.

6 M. B. Giles. "Improved multilevel Monte Carlo convergence using the Milstein scheme," p.343-358 in Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer, 2008.

7 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.

8 A. B. Owen. "A randomized Halton algorithm in R2017." arXiv:1706.02808 [stat.CO].

9 P. L'Ecuyer, P. Marion, M. Godin, and F. Puchhammer, "A Tool for Custom Construction of QMC and RQMC Point Sets," Monte Carlo and Quasi-Monte Carlo Methods 2020.