# (Quasi-)Monte Carlo Importance Sampling with QMCPy

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

[1]Illinois Institute of Technology (IIT), Department of Applied Mathematics
[2]Kamakura Corporation
[3]SigOpt, an Intel company
[4]Wi-Tronix LLC

February 24, 2021

# (Q)MC Math

Applications in applied statistics, finance, computer graphics, ...

$$\mu = \int_{\mathcal{T}} g(\boldsymbol{t})\lambda(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} = \int_{[0,1]^d} g(\boldsymbol{\Psi}(\boldsymbol{x}))\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}[f(\boldsymbol{X})]$$

$$\boldsymbol{X} \sim \mathcal{U}[0,1]^d$$

Original Integrand $g : \mathcal{T} \to \mathbb{R}$

True Measure $\lambda : \mathcal{T} \to \mathbb{R}^+$ e.g. probability density or 1 for Lebesgue measure

Transformation $\boldsymbol{\Psi} : [0,1]^d \to \mathcal{T}$ with Jacobian $|\boldsymbol{\Psi}'(\boldsymbol{x})|$

Transformed Integrand $f : [0,1]^d \to \mathbb{R}$

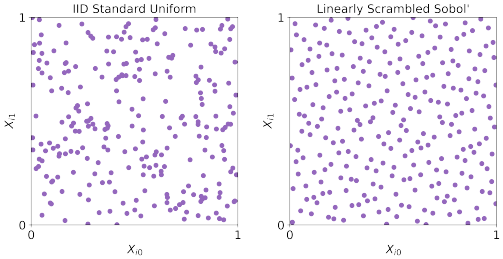
# (Q)MC Approximation

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

Discrete Distribution: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_n \sim \mathcal{U}[0,1]^d$

Monte Carlo: $\{\boldsymbol{x}_i\}_{i=1}^n$ IID (Independent Identically Distributed)

Quasi-Monte Carlo: $\{\boldsymbol{x}_i\}_{i=1}^n$ LD (Low-Discrepancy)



# Standard Keister Example [1]

$$
\begin{aligned}
\mu &= \int_{\mathbb{R}^d} \cos(\|\boldsymbol{t}\|) \exp(-\|\boldsymbol{t}\|^2) \, \mathrm{d}\boldsymbol{t} \\
&= \int_{\mathbb{R}^d} \underbrace{\pi^{d/2} \cos(\|\boldsymbol{t}\|)}_{g(\boldsymbol{t})} \underbrace{\pi^{-d/2} \exp(-\|\boldsymbol{t}\|^2)}_{\lambda(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t} \mid \boldsymbol{0}, \mathrm{I}/2)} \, \mathrm{d}\boldsymbol{t} \\
&= \int_{[0,1]^d} \pi^{d/2} \cos(\|\boldsymbol{\Psi}(\boldsymbol{x})\|) \, \mathrm{d}\boldsymbol{x} \\
&= \int_{[0,1]^d} \underbrace{g(\boldsymbol{\Psi}(\boldsymbol{x}))}_{f(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x}
\end{aligned}
$$

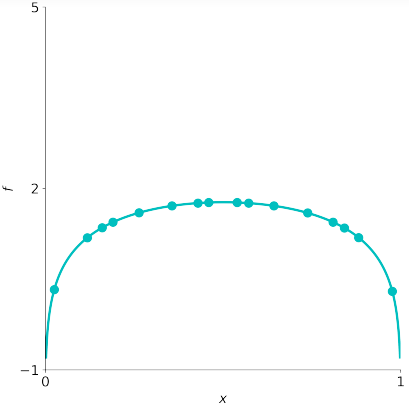
Where $\boldsymbol{\Psi}(\boldsymbol{x}) = \boldsymbol{\Phi}^{-1}(\boldsymbol{x})/\sqrt{2}$ and $\boldsymbol{\Phi}$ is the standard normal CDF.
Chose $\boldsymbol{\Psi}$ so $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$ and $f(\boldsymbol{x}) = g(\boldsymbol{\Psi}(\boldsymbol{x}))$, weight and Jacobian cancel

Import QMCPy [2] and NumPy [3]

```
>>> import qmcpy as qp
>>> import numpy as np
```

Sample the Standard Keister

```
>>> sobol = qp.Sobol(dimension=3,seed=11)
>>> keister = qp.Keister(sobol)
>>> x = sobol.gen_samples(2**4)
>>> y = keister.f(x)
```

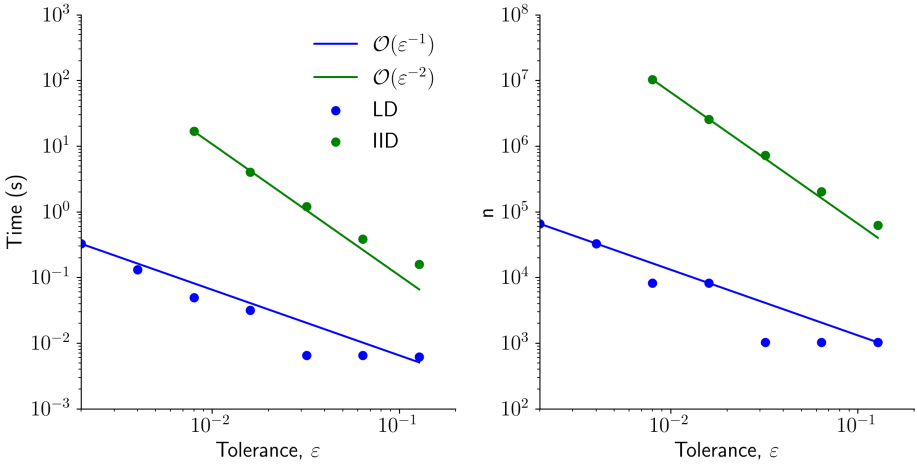



Numerical integration based on `CubQMCSobolG` [4] stopping criterion

```
>>> stop_obj = qp.CubQMCSobolG(keister,abs_tol=5e-6)
>>> solution,data = stop_obj.integrate()
```

```
>>> data # print(data)
Solution: 2.1683
Keister (Integrand Object)
Sobol (DiscreteDistribution Object)
    d               3
    randomize       1
    seed            11
    mimics          StdUniform
Gaussian (TrueMeasure Object)
    mean            0
    covariance      2^(-1)
CubQMCSobolG (StoppingCriterion Object)
    abs_tol         5.00e-06
    rel_tol         0
LDTransformData (AccumulateData Object)
    n_total         2^(22)
    solution        2.168
    error_bound     3.36e-06
    time_integrate  3.769
```

# Standard Keister Integrand in 5 Dimensions



# (Q)MC with Importance Sampling (IS)

Standard (Q)MC: select $\boldsymbol{\Psi}$ so that $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$

(Q)MC with IS: select $\boldsymbol{\Psi}_{\mathsf{IS}}$ so that $\lambda_{\mathsf{IS}}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))|\boldsymbol{\Psi}'_{\mathsf{IS}}(\boldsymbol{x})| = 1$

$$\begin{aligned}\mu = \int_{\mathcal{T}} g(\boldsymbol{t})\lambda(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} &= \int_{[0,1]^d} g(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))\lambda(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))|\boldsymbol{\Psi}'_{\mathsf{IS}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \\ &= \int_{[0,1]^d} \underbrace{g(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))\frac{\lambda(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))}{\lambda_{\mathsf{IS}}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))}}_{f_{\mathsf{IS}}(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

Select $\lambda_{\mathsf{IS}} : \mathcal{T} \to \mathbb{R}^+$ so that $\mathrm{Var}(f_{\mathsf{IS}})$ is small.

More expensive to evaluate $f_{\mathsf{IS}}$ than $f(\boldsymbol{x}) = g(\boldsymbol{\Psi}(\boldsymbol{x}))$

# Gaussian Keister IS Example

$$\mu = \int_{\mathbb{R}^d} \underbrace{\pi^{d/2}\cos(\|\boldsymbol{t}\|)}_{g(\boldsymbol{t})} \underbrace{\pi^{-d/2}\exp(-\|\boldsymbol{t}\|^2)}_{\lambda(\boldsymbol{t})} \,\mathrm{d}\boldsymbol{t}$$

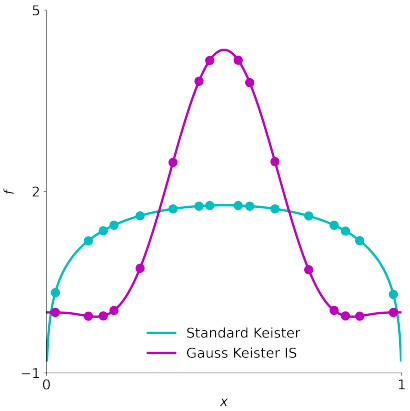
Suppose we choose $\boldsymbol{\Psi}_{\mathsf{IS}}$ so $\lambda_{\mathsf{IS}}(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{\mu}, \Sigma)$ and $\lambda_{\mathsf{IS}}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))|\boldsymbol{\Psi}'_{\mathsf{IS}}(\boldsymbol{x})| = 1$

$$\begin{aligned}\mu &= \int_{[0,1]^d} \pi^{d/2}\cos(\|\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})\|)\mathcal{N}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})|\mathbf{0}, \mathsf{I}/2)\,|\boldsymbol{\Psi}'_{\mathsf{IS}}(\boldsymbol{x})|\,\mathrm{d}\boldsymbol{x} \\ &= \int_{[0,1]^d} \pi^{d/2}\cos(\|\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})\|)\,\frac{\mathcal{N}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})|\mathbf{0}, \mathsf{I}/2)}{\mathcal{N}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})|\boldsymbol{\mu}, \Sigma)}\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

If $\lambda_{\mathsf{IS}}(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\mathbf{0}, \mathsf{I}/2)$, then we recover $f_{\mathsf{IS}}(\boldsymbol{x}) = f(\boldsymbol{x}) = \pi^{d/2}\cos(\|\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})\|)$

# Keister Integrand Importance Sampled by $\lambda_{\text{IS}}(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{0}, 3\text{I})$

```
>>> sobol = qp.Sobol(dimension=3,seed=11)
>>> transform = qp.Gaussian(sobol,
...     mean = 0,
...     covariance = 3)
>>> keister_gauss = qp.Keister(transform)
>>> stop_obj = qp.CubQMCSobolG(
...     integrand = keister_gauss,
...     abs_tol = 5e-6)
>>> solution,data = stop_obj.integrate()
```




Took 70% the time and 50% the samples compared to standard Keister

```
>>> data
Solution: 2.1683
Gaussian (TrueMeasure Object)
    mean            0
    covariance      2^(-1)
    decomp_type     pca
    transform       Gaussian (TrueMeasure Object)
                       mean            0
                       covariance      3
                       decomp_type     pca
LDTransformData (AccumulateData Object)
    n_total         2^(21)
    solution        2.168
    error_bound     2.47e-06
    time_integrate  2.635
```

# Asian Option Example

Time Horizon $\tau$; Interest Rate $r$; Volatility $\sigma$; Start Price $S_0$; Strike Price $K$

True Measure

$$\boldsymbol{T} = (T_1, \ldots, T_d) \sim \mathcal{N}(\boldsymbol{0}, \Sigma) \qquad \text{for} \qquad \Sigma = (\tau/d)(\min(k,j))_{j,k=1}^{d}$$

Asset Path

$$S_j(\boldsymbol{T}) = S_0 \exp((r - \sigma^2/2)\tau j/d + \sigma T_j) \qquad \text{for} \qquad j = 1, \ldots, d$$

Payoff

$$\text{Pay}(\boldsymbol{S}) = \max\left(\frac{1}{2d}\sum_{j=1}^{d}(S_{j-1} + S_j) - K, 0\right) \qquad \text{for} \qquad \boldsymbol{S} = (S_0, \ldots, S_d)$$

Expected Discounted Payoff

$$\mu = \mathbb{E}(g(\boldsymbol{T})) = \int_{\mathbb{R}^d} g(\boldsymbol{t})\mathcal{N}(\boldsymbol{t}|\boldsymbol{0}, \Sigma)\,\mathrm{d}\boldsymbol{t} \qquad \text{for} \qquad g(\boldsymbol{t}) = \text{Pay}(\boldsymbol{S}(\boldsymbol{t}))\exp(-r\tau)$$

**Standard (Q)MC:** Choose $\boldsymbol{\Psi}$ so that $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$

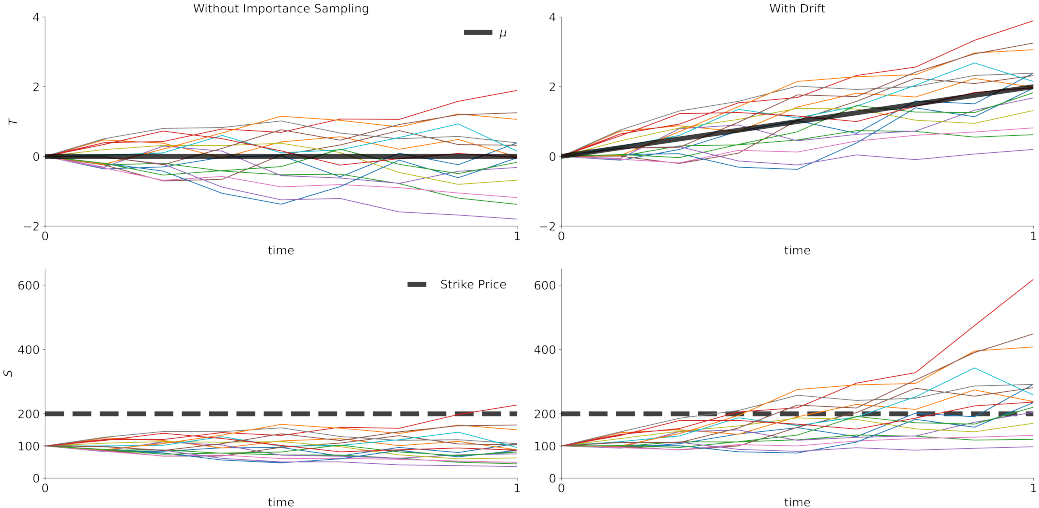
$$\mu = \int_{\mathbb{R}^d} g(\boldsymbol{t})\mathcal{N}(\boldsymbol{t}|\boldsymbol{0},\Sigma)\,\mathrm{d}\boldsymbol{t} = \int_{[0,1]} g(\boldsymbol{\Psi}(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x} = \int_{[0,1]} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$$

**Importance Sampling** with drift $\alpha$: $\lambda_{\mathsf{IS}}(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{\alpha},\Sigma)$ for $\boldsymbol{\alpha} = \alpha(\tau j/d)_{j=1}^d$
Choose $\boldsymbol{\Psi}_{\mathsf{IS}}$ so that $\lambda_{\mathsf{IS}}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))|\boldsymbol{\Psi}_{\mathsf{IS}}'(\boldsymbol{x})| = 1$

$$\mu = \int_{\mathbb{R}^d} g(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))\,\frac{\mathcal{N}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})|\boldsymbol{0},\Sigma)}{\mathcal{N}(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x})|\boldsymbol{\alpha},\Sigma)}\,\mathrm{d}\boldsymbol{x} = \int_{[0,1]^d} g(\boldsymbol{\Psi}_{\mathsf{IS}}(\boldsymbol{x}))\exp(\alpha(\alpha\tau/2 - \boldsymbol{\Psi}_{\mathsf{IS},d}(\boldsymbol{x})))\,\mathrm{d}\boldsymbol{x}$$

Since $\Sigma^{-1}\boldsymbol{\alpha} = (0,0,\ldots,0,a)^T$

Time Horizon $\tau = 1$; Interest Rate $r = 0$; Volatility $\sigma = 0.5$;
Start Price $S_0 = 100$; Strike Price $K = 200$; Drift $\alpha = 2$



```
>>> sobol = qp.Sobol(dimension=8,seed=7)
>>> params = {
...     "start_price":100,
...     "strike_price":200,
...     "call_put":"call"}
>>> # Without IS
>>> aco = qp.AsianOption(sobol,**params)
>>> _,data = qp.CubQMCSobolG(aco,abs_tol=1e-4).integrate()
>>> # With Drift IS
>>> bmIS = qp.BrownianMotion(sobol,drift=2)
>>> acoIS = qp.AsianOption(bmIS,**params)
>>> _,dataIS = qp.CubQMCSobolG(acoIS,abs_tol=1e-4).integrate()
```

| | Time (sec) | | Samples | |
|---|---|---|---|---|
| $\varepsilon$ | $1e-4$ | $1e-5$ | $1e-4$ | $1e-5$ |
| Without IS | 2.7 | 55.2 | 2.1e6 | 3.4e7 |
| With Drift IS | 1.1 (42%) | 23.7 (42%) | 5.2e5 (25%) | 8.4e6 (25%) |

```
>>> data
Solution: 0.1759
BrownianMotion (TrueMeasure Object)
    time_vec        [0.125 0.25 0.375 ... ]
    drift           0
    mean            [0.    0.   0.    ... ]
    covariance      ...
>>> dataIS
Solution: 0.1760
BrownianMotion (TrueMeasure Object)
    time_vec        [0.125 0.25  0.375 ... ]
    drift           0
    mean            [0.    0.    0.    ... ]
    covariance      ...
    transform       BrownianMotion (TrueMeasure Object)
                       time_vec        [0.125 0.25  0.375 ... ]
                       drift           2^(1)
                       mean            [0.25  0.5   0.75  ... ]
                       covariance      ...
```

# Composed IS

For $L$ composed transforms, let $\boldsymbol{\Psi}_0(\boldsymbol{x}) = \boldsymbol{x}$ and define

$$\text{the first } \ell \text{ transforms as } \hat{\boldsymbol{\Psi}}_\ell = (\boldsymbol{\Psi}_\ell \circ \boldsymbol{\Psi}_{\ell-1} \circ \cdots \circ \boldsymbol{\Psi}_0) \text{ and}$$
$$\text{the complete transform as } \hat{\boldsymbol{\Psi}}_L = (\boldsymbol{\Psi}_L \circ \boldsymbol{\Psi}_{L-1} \circ \cdots \circ \boldsymbol{\Psi}_0) = \boldsymbol{\Psi}_{\mathsf{IS}}.$$

If $\lambda_\ell(\boldsymbol{\Psi}_\ell(\boldsymbol{x}))|\boldsymbol{\Psi}_\ell'(\boldsymbol{x})| = 1$ for $\ell = 1, \dots, L$, then

$$\begin{aligned}
\mu &= \int_{\mathcal{T}} g(\boldsymbol{t})\lambda(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t} \\
&= \int_{[0,1]^d} g(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x}))\lambda(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x})) \prod_{\ell=1}^{L} |\boldsymbol{\Psi}_\ell'(\hat{\boldsymbol{\Psi}}_{\ell-1}(\boldsymbol{x}))|\,\mathrm{d}\boldsymbol{x} \\
&= \int_{[0,1]^d} g(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x})) \frac{\lambda(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x}))}{\prod_{\ell=1}^{L} \lambda_\ell(\hat{\boldsymbol{\Psi}}_\ell(\boldsymbol{x}))}\,\mathrm{d}\boldsymbol{x}
\end{aligned}$$

# Gauss-Kuma Keister IS

Compose

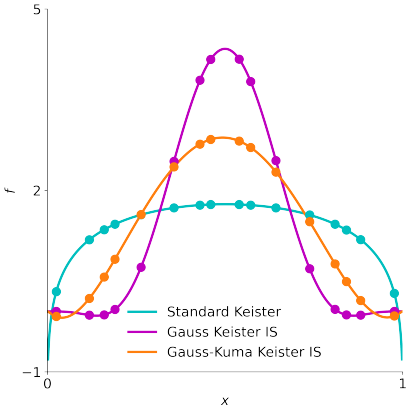
$\boldsymbol{\Psi}_1 : [0,1]^d \to [0,1]^d$, the inverse CDF of Kumaraswaramy $\mathcal{K}(\boldsymbol{a}, \boldsymbol{b})$, and
$\boldsymbol{\Psi}_2 : [0,1]^d \to \mathbb{R}^d$, the Gaussian $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ transform, so that

$$\mathcal{K}(\boldsymbol{\Psi}_1(\boldsymbol{x})|\boldsymbol{a}, \boldsymbol{b})|\boldsymbol{\Psi}_1'(\boldsymbol{x})| = 1 = \mathcal{N}(\boldsymbol{\Psi}_2(\boldsymbol{x})|\boldsymbol{\mu}, \Sigma)|\boldsymbol{\Psi}_2'(\boldsymbol{x})|.$$

Then

$$\begin{aligned}\mu &= \int_{\mathbb{R}^d} \pi^{d/2} \cos(\|\boldsymbol{t}\|) \mathcal{N}(\boldsymbol{t}, |\boldsymbol{0}, \mathsf{I}/2) \, \mathrm{d}\boldsymbol{t} \\ &= \int_{[0,1]^d} \pi^{d/2} \cos(\|(\boldsymbol{\Psi}_2 \circ \boldsymbol{\Psi}_1)(\boldsymbol{x})\|) \frac{\mathcal{N}((\boldsymbol{\Psi}_2 \circ \boldsymbol{\Psi}_1)(\boldsymbol{x})|\boldsymbol{0}, \mathsf{I}/2)}{\mathcal{N}((\boldsymbol{\Psi}_2 \circ \boldsymbol{\Psi}_1)(\boldsymbol{x})|\boldsymbol{\mu}, \Sigma) \mathcal{K}(\boldsymbol{\Psi}_1(\boldsymbol{x})|\boldsymbol{a}, \boldsymbol{b})} \, \mathrm{d}\boldsymbol{x}\end{aligned}$$

```
>>> sobol = qp.Sobol(dimension=3,seed=11)
>>> tf1_kuma = qp.Kumaraswamy(
...     sampler = sobol,
...     a = .8,
...     b = .8)
>>> tf2_gauss = qp.Gaussian(
...     sampler = tf1_kuma)
>>> keister_kuma_gauss = qp.Keister(
...     sampler = tf2_gauss)
>>> stop_obj = qp.CubQMCSobolG(
...     integrand = keister_kuma_gauss,
...     abs_tol = 5e-6)
>>> solution,data = stop_obj.integrate()
```

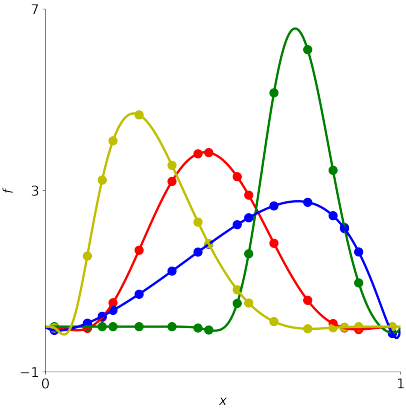



Took 35% the time and 25% the samples compared to standard Keister

```
>>> data
Solution: 2.1683
Gaussian (TrueMeasure Object)
    mean            0
    covariance      2^(-1)
    decomp_type     pca
    transform       Gaussian (TrueMeasure Object)
                       mean            0
                       covariance      1
                       decomp_type     pca
                       transform       Kumaraswamy (TrueMeasure
                                          a                0.800
                                          b                0.800
LDTransformData (AccumulateData Object)
    n_total         2^(20)
    solution        2.168
    error_bound     2.25e-06
    time_integrate  1.320
```

# QMCPy Resources

- PyPI: pypi.org/project/qmcpy/
- GitHub: github.com/QMCSoftware/QMCSoftware
- Documentation: qmcpy.readthedocs.io
- Blogs: qmcpy.org
- MCQMC2020 Tutorial
  - Slides: qmcpy.org/mcqmc-2020-tutorial/
  - Notebook: tinyurl.com/QMCPyTutorial
  - "Quasi-Monte Carlo Software" Article [5]



# References


1. Keister, B. D. Multidimensional Quadrature Algorithms. *Computers in Physics* **10,** 119–122 (1996).
2. Choi, S.-C. T., Hickernell, F. J., McCourt, M. & Sorokin, A. *QMCPy: A quasi-Monte Carlo Python Library*. 2020+. https://github.com/QMCSoftware/QMCSoftware.
3. Oliphant, T. *Guide to NumPy*. https://ecs.wgtn.ac.nz/foswiki/pub/Support/ManualPagesAndDocumentation/numpybook.pdf (Trelgol Publishing USA, 2006).
4. Hickernell, F. J. & Jiménez Rugama, L. A. *Reliable Adaptive Cubature Using Digital Sequences*. 2014. arXiv: 1410.8615 [math.NA].
5. Choi, S.-C. T., Hickernell, F. J., Jagadeeswaran, R., McCourt, M. J. & Sorokin, A. G. *Quasi-Monte Carlo Software*. 2021. arXiv: 2102.07833 [cs.MS].