(Quasi-)Monte Carlo Importance Sampling with QMCPy

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

[1]Department of Applied Mathematics, Illinois Institute of Technology, RE 220, 10 W. 32nd St., Chicago, IL 60616.; [2]Kamakura Corporation, 2222 Kalakaua Ave, Suite 1400, Honolulu, HI 96815.; [3]SigOpt, an Intel company, 100 Bush St., Suite 1100, San Francisco, CA 94104.; [4]Wi-Tronix LLC, 631 E Boughton Rd, Suite 240, Bolingbrook, IL 60440

(Quasi-)Monte Carlo, (Q)MC, methods are a class of powerful numerical integration algorithms that have been proven to scale well to high dimensions. Various techniques exist to decrease the computational cost of (Q)MC methods. This article focuses on importance sampling, a technique that performs variable transformations to make the integral easier for (Q)MC approximation. The build up to composed importance sampling is paralleled by code from our QMCPy package that implements these concepts.

Quasi-Monte Carlo | Importance Sampling | Numerical Software

Monte Carlo methods have become science's sharpest tool for high dimensional numerical integration. In low dimensions, using Gaussian quadrature, Simpson's rule, or similar numerical methods will provide fast, accurate approximation. However, as the dimension of the problem grows, these methods quickly become computationally intractable. By introducing randomness, Monte Carlo methods are able to approximate high dimensional integrals much more efficiently than the aforementioned techniques.

The central idea of Monte Carlo integration is to view an integral as the expectation of a function of a random variable with a *well-defined* probability distribution. Samples may then be drawn from that probability distribution, and the sample average of the corresponding function evaluations is the Monte Carlo integral estimate. The form of the integral that is amenable to Monte Carlo integration may require applying a change of variables to the original integral.

Monte Carlo methods can be improved with two central ideas: smarter sampling and rewriting the integrand. While standard Monte Carlo (MC) methods sample the probability distribution at independent nodes, Quasi-Monte Carlo (QMC) methods carefully coordinate sampling locations to achieve significantly faster convergence to the true mean. QMC methods can be straightforward to adapt to your standard MC problem, often requiring little more than replacing independent samples with low-discrepancy sequences.

Often times, the integrand can be rewritten to mitigate sharp peaks and valleys while preserving the value of the integral. These smoothing transforms make the integrand easier for (Q)MC methods to approximate. Control variates and importance sampling are among the most effective rewriting methods, although the choice of good control variate functions and importance sampling measures is currently more of an art than a science.

In this article, we focus on importance sampling and an extension to support multiple measures in a framework we call composed importance sampling. We have implemented this framework into QMCPy (1), our community developed

Significance Statement

This work discusses the mathematics and implementation of importance sampling for (Quasi-)Monte Carlo methods. We extend the standard development to accommodate multiple measures in a composed importance sampling framework. This framework is demonstrated using our implementation in QMCPy, a community driven Quasi-Monte Carlo software.

Quasi-Monte Carlo Library in Python 3. Python was chosen to help make QMCPy easily accessible to the community and extendable by decades of research in (Q)MC. Throughout this work, we use QMCPy to demonstrate the benefits of importance sampling through a running example from Keister (2). Those interested in following the development of QMCPy should visit qmcpy.org.

The remainder of this article is organized as follows. We first present the (Q)MC problem and differentiate between MC and QMC methods. We then generalize the (Q)MC problem to incorporate importance sampling and extend this notion to composed importance sampling. We end by summarizing developments, discussing future research, and linking to additional resources for the QMCPy package.

(Quasi-)Monte Carlo Methods

The model problem for (Q)MC takes the form of

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

where $g : \mathcal{T} \to \mathbb{R}$ is the *original integrand* and $\lambda : \mathcal{T} \to \mathbb{R}^+$ is a non-negative weight function which we call the *true measure*. Often times the true measure is a probability distribution, but we are not restricted to this setting. For instance, the Lebesgue measure can be used by setting $\lambda(\boldsymbol{t}) = 1$.

In order to perform (Q)MC simulation, we rewrite Eq. [1] as the d dimensional integrand

$$\mu = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \qquad [2]$$

where the *transformed integrand* is

$$f(\boldsymbol{x}) = g(\boldsymbol{\Psi}(\boldsymbol{x}))\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})|. \qquad [3]$$

The change of variables is captured in $\boldsymbol{\Psi} : [0,1]^d \to \mathcal{T}$, the *transform*, and is central to importance sampling. We denote the Jacobian determinant of this transform by $|\boldsymbol{\Psi}'(\boldsymbol{x})|$. Eq. [2] can be viewed as taking the expectation of $f(\boldsymbol{X})$ when $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$.

(Q)MC methods approximate the true mean μ by the sample mean of f evaluated at sampling nodes $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \sim \mathcal{U}[0,1]^d$. We denote the n-sample mean by $\hat{\mu}_n$ so that

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

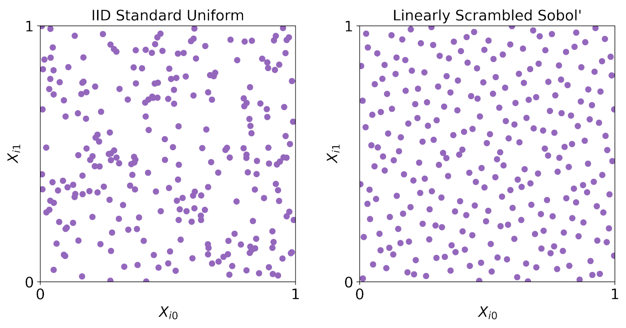

Fig. 1. IID points contrasted with LD Sobol' points, both mimicking $\mathcal{U}[0,1]^2$. Note how the LD sequence covers the domain more evenly than the IID points.

Standard MC methods select $\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$ to be independent and identically distributed (IID). Recall that the multivariate probability distribution of n IID points is the product of the marginals: $F_n(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = \prod_{i=0}^{n-1} F(\boldsymbol{x}_i)$. If $\boldsymbol{X}$ is a standard uniform random variable, then $F(\boldsymbol{x}) = x_1 \cdots x_d$ for $\boldsymbol{x} := (x_1, \ldots, x_d) \in [0,1]^d$.

QMC methods carefully select sampling nodes $\{\boldsymbol{X}_i\}_{i=0}^{n-1}$ so that their empirical distribution mimics F better than the empirical distribution of IID points. The discrepancy is a measure of the difference between an empirical distribution and a target distribution. Therefore, the sampling nodes used by QMC methods are called *low-discrepancy* (LD) node sets. Common examples of LD node sets are digital sequences, integration lattices, and Halton points. Figure 1 contrasts IID and LD node sets that mimic $\mathcal{U}[0,1]^2$. Notice how the IID points leave gaps and clusters while the LD sequence covers the domain more uniformly. This better uniformity, or lower discrepancy, enables QMC integral approximation to converge significantly faster than standard MC methods.

Importance Sampling

When true measure λ corresponds to a probability density, we can often select a transform $\boldsymbol{\Psi}$ so that $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$, i.e. the weight and Jacobian cancel. In this case, Eq. [3] simplifies to $f = g \circ \boldsymbol{\Psi}$, which is relatively easy to evaluate. However, it may be necessary or advantageous to choose $\boldsymbol{\Psi}$ so the weight and Jacobian do not cancel. For instance, no canceling transform exists when λ corresponds to a Lebesgue measure on $\mathbb{R}^d$. As we will show later, selecting $\boldsymbol{\Psi}$ so $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| \neq 1$ can often speed up a (Q)MC approximation even if a canceling transform does exist.

Importance sampling tries to select a transform $\boldsymbol{\Psi}$ to better sample the original integrand in places of higher variation. In doing so, the transformed integrand has lower variation and is therefore easier for (Q)MC methods to approximate. Specifically, anytime $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| \neq 1$, we are performing importance sampling. In this case, we say $\boldsymbol{\Psi}(\boldsymbol{X})$ is stochastically equivalent to a random variable with density $\tilde{\lambda}$ when $\boldsymbol{X} \sim U[0,1]^d$. This implies $\tilde{\lambda}(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$ for some $\tilde{\lambda} : \mathcal{T} \to \mathbb{R}^+$ so that

$$f(\boldsymbol{x}) = g(\boldsymbol{\Psi}(\boldsymbol{x}))\,\frac{\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))}{\tilde{\lambda}(\boldsymbol{\Psi}(\boldsymbol{x}))}. \qquad [4]$$

The choice of $\boldsymbol{\Psi}$, or equivalently $\tilde{\lambda}$, is currently a manual task that often requires a good deal of problem-specific knowledge.

The Keister Integrand. Suppose we want to approximate the following integrand from Keister

$$\mu = \int_{\mathbb{R}^d} \cos(\|\boldsymbol{t}\|)\exp(-\|\boldsymbol{t}\|^2)\mathrm{d}\boldsymbol{t}. \qquad [5]$$

Notice that we may split the integrand into original function g and true measure λ so that

$$\mu = \int_{\mathbb{R}^d} \underbrace{\pi^{d/2}\cos(\|\boldsymbol{t}\|)}_{g(\boldsymbol{t})}\,\underbrace{\pi^{-d/2}\exp(-\|\boldsymbol{t}\|^2)}_{\lambda(\boldsymbol{t})}\,\mathrm{d}\boldsymbol{t}, \qquad [6]$$

and λ is the probability density of a multivariate Gaussian with mean $\mathbf{0}$ and covariance $\mathsf{I}/2$. We denote the density of this true measure by $\lambda(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\mathbf{0}, \mathsf{I}/2)$.

To approximate this integral, we use QMCPy, a community-develop QMC package for Python. QMCPy includes the Keister integrand as just defined. To set up the (Q)MC problem without importance sampling, we only need to define a LD sequence. In this case, we choose the LD Sobol' sequence (see Figure 1) in order to perform QMC quadrature.

First, we import QMCPy via:

```
>>> import qmcpy as qp
```

Then, we initialize a 3-dimensional Sobol' sequence and Keister integrand:

Listing 1. Standard Keister Construction

```
>>> d = 3
>>> sobol = qp.Sobol(d,seed=11)
>>> K_std = qp.Keister(sobol)
```

The transform defaults to $\boldsymbol{\Psi}(\boldsymbol{x}) = \boldsymbol{\Phi}^{-1}(\boldsymbol{x})/\sqrt{2}$, where $\boldsymbol{\Phi}^{-1}$ is the element-wise inverse CDF of a standard Gaussian. This transform was chosen so that $\lambda(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$, and we recover $f = g \circ \boldsymbol{\Psi}$. Therefore, our standard, or default, Keister function is *not* using importance sampling.

To evaluate the integrand, we first generate samples from the Sobol' sequence. These samples are then input to the transformed integrand, f, which was automatically constructed for us by QMCPy.

```
>>> x = sobol.gen_samples(2**4)
>>> y = K_std.f(x)
```

Figure 2 shows the standard Keister integrand in one-dimension evaluated at the first 2^4 points of a scrambled Sobol' sequence.

We may now run a QMC quadrature algorithm to approximate the integrand to within absolute tolerance $\epsilon = 5\text{e-}6$.

Listing 2. Stopping Criterion Evaluation

```
>>> sc = qp.CubQMCSobolG(
...      integrand = K_std,
...      abs_tol = 5e-6)
>>> sol,data = sc.integrate()
```

In this case the *stopping criterion* algorithm, which determines the number of samples necessary to achieve the desired tolerance, is based on the decay of the integrand's Fourier Walsh coefficients. The outputs of calling `integrate()` on the `qp.CubQMCSobolG` (3) stopping criterion are a numerical solution and data object that houses integration information. Table 1 collects the number of samples and run time from this data object.

We now turn our attention to importance sampling the Keister integrand. For this example, we choose transform $\boldsymbol{\Psi}(\boldsymbol{x}) = \sqrt{3}\boldsymbol{\Phi}^{-1}$ so that $\tilde{\lambda}(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{0}, 3\mathsf{I})$ and $\tilde{\lambda}(\boldsymbol{\Psi}(\boldsymbol{x}))|\boldsymbol{\Psi}'(\boldsymbol{x})| = 1$. Therefore, our transformed Keister integrand becomes

$$f(\boldsymbol{x}) = \pi^{d/2}\cos(\|\boldsymbol{\Psi}(\boldsymbol{x})\|)\,\frac{\mathcal{N}(\boldsymbol{\Psi}(\boldsymbol{x})|\boldsymbol{0},\mathsf{I}/2)}{\mathcal{N}(\boldsymbol{\Psi}(\boldsymbol{x})|\boldsymbol{0},3\mathsf{I})}.$$

In QMCPy, we first define our importance sampling measure, in this case $\mathcal{N}(\boldsymbol{0}, 3\mathsf{I})$, and then use it to construct our Keister integrand with an importance `sampler`.

```
>>> transform = qp.Gaussian(
...     sampler = sobol,
...     mean = 0,
...     covariance = 3)
>>> K_gauss = qp.Keister(transform)
```

Note how we construct the `Keister` integrand using the `Gaussian` importance sampling measure rather than the `sobol` generator as done in Listing 1. A one-dimensional version of this importance sampling integrand is plotted in Figure 2 as `Gauss Keister IS`. While the one and three dimensional Keister examples give similar savings for this example, it is not always the case that copying parameters to different dimensions will result in similar success.

QMC integration of the above Keister function with importance sampling is done in an analogous manner to Listing 2. Table 1 compares the required time and samples against the non-importance sampled Keister. Notice how the savings in samples is greater than the savings in time since the importance sampling integrand requires an additional Jacobian computation at each sample.

Composed Importance Sampling

We now generalize importance sampling to allow multiple sub-transforms. Let $\boldsymbol{\Psi}_0(\boldsymbol{x}) = \boldsymbol{x}$ and denote the composition of the first ℓ (non-identity) sub-transforms by $\hat{\boldsymbol{\Psi}}_\ell = (\boldsymbol{\Psi}_\ell \circ \boldsymbol{\Psi}_{\ell-1} \circ \cdots \circ \boldsymbol{\Psi}_0)$. Therefore, the complete L sub-transform composition is $\boldsymbol{\Psi} = \hat{\boldsymbol{\Psi}}_L$. The sub-transforms must be compatible with the discrete distribution and true measure, meaning $\boldsymbol{\Psi}_\ell : [0,1]^d \to [0,1]^d$ for $\ell = 1, \dots L-1$ and $\boldsymbol{\Psi}_L : [0,1]^d \to \mathcal{T}$. Define λ_l to be the density of $\boldsymbol{\Psi}_\ell(\boldsymbol{X}_\ell)$ for $\boldsymbol{X}_\ell \sim \mathcal{U}[0,1]^d$ so that $\lambda_\ell(\boldsymbol{\Psi}_\ell(\boldsymbol{x}))|\boldsymbol{\Psi}'_\ell(\boldsymbol{x})| = 1$. The chain rule then implies that

$$\begin{aligned} f(\boldsymbol{x}) &= g(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x}))\,\lambda(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x}))\prod_{\ell=1}^{L}|\boldsymbol{\Psi}'_\ell(\hat{\boldsymbol{\Psi}}_{\ell-1}(\boldsymbol{x}))| \\ &= g(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x}))\,\frac{\lambda(\hat{\boldsymbol{\Psi}}_L(\boldsymbol{x}))}{\prod_{\ell=1}^{L}\lambda_\ell(\hat{\boldsymbol{\Psi}}_\ell(\boldsymbol{x}))}. \end{aligned}$$

The Keister Integrand. Let us now return to the Keister integrand which is defined in Eq. [6] with original integrand $g(\boldsymbol{t}) = \pi^{d/2}\cos(\|\boldsymbol{t}\|)$ and true measure $\lambda(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{0},\mathsf{I}/2)$. Suppose we are interested in importance sampling by a composed Gaussian-Kumaraswamy distribution. The Kumaraswamy distribution (4), denoted $\mathcal{K}(\boldsymbol{a},\boldsymbol{b})$, may be sampled via an inverse CDF transform $\boldsymbol{\Psi}_K : [0,1]^d \to [0,1]^d$, and the Gaussian distribution, denoted $\mathcal{N}(\boldsymbol{\mu}, \Sigma = \mathsf{A}\mathsf{A}^T)$, may be sampled via transform $\boldsymbol{\Psi}_G : [0,1]^d \to \mathbb{R}^d$ defined as $\boldsymbol{\Psi}_G(\boldsymbol{x}) = \mathsf{A}\boldsymbol{\Phi}^{-1} + \boldsymbol{\mu}$. Again, $\boldsymbol{\Phi}^{-1}$ is the element-wise inverse CDF transform of a standard normal. Due to the nature of these transforms, we have $\lambda_K(\boldsymbol{t}) = \mathcal{K}(\boldsymbol{t}|\boldsymbol{a},\boldsymbol{b})$, $\lambda_G(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{\mu},\Sigma)$, and $\lambda_K(\boldsymbol{\Psi}_K(\boldsymbol{x}))|\boldsymbol{\Psi}'_K(\boldsymbol{x})| = 1 = \lambda_G(\boldsymbol{\Psi}_G(\boldsymbol{x}))|\boldsymbol{\Psi}'_G(\boldsymbol{x})|$. Therefore, we define the complete transform as $\boldsymbol{\Psi} = \boldsymbol{\Psi}_G \circ \boldsymbol{\Psi}_K$ so that

$$f(\boldsymbol{x}) = \frac{\pi^{d/2}\cos(\|\boldsymbol{\Psi}(\boldsymbol{x})\|)\,\mathcal{N}(\boldsymbol{\Psi}(\boldsymbol{x})|\boldsymbol{0},\mathsf{I}/2)}{\mathcal{K}(\boldsymbol{\Psi}_K(\boldsymbol{x})|\boldsymbol{a},\boldsymbol{b})\,\mathcal{N}(\boldsymbol{\Psi}(\boldsymbol{x})|\boldsymbol{\mu},\Sigma)}.$$

In QMCPy, we compose transforms through the nested construction of measure objects. The following code uses the `sobol` instance from Listing 1 to construct the first Kumaraswamy transform which is used to construct the second Gaussian transform. Note how the construction of

Table 1. Comparing the time (seconds) and samples necessary to integrate Keister functions to within absolute tolerance 5e-6. The percent usage of importance sampling (IS) integrands compared to the standard, non-importance sampling, integrand are also displayed. Using importance sampling significantly decreases the computational budget required for accurate approximation. The Sobol sequence and corresponding stopping criterion favor sample sizes that are powers of 2.

	Time	Samples
Standard Keister	3.8	2^{22}
Gauss Keister IS	2.6 (70%)	2^{21} (50%)
Gauss-Kuma Keister IS	1.3 (35%)	2^{20} (25%)

QMCPy objects reflects the composition of subtransforms used to define the complete transform. Finally, the Gaussian-Kumaraswamy measure is used to construct a Keister integrand which defines the original integrand and true measure.

```
>>> tf_K = qp.Kumaraswamy(
...     sampler = sobol,
...     a = .8,
...     b = .8)
>>> tf_G = qp.Gaussian(
...     sampler = tf_K)
>>> K_Gauss_Kuma = qp.Keister(tf_G)
```

Figure 2 plots a one-dimensional `Gauss-Kuma Keister IS` function against the Keister functions developed previously. Again, we may integrate the composed importance sampling measure using Listing 2. The results of this integration are displayed in Table 1. Both importance sampling examples deliver significant savings in time and samples compared to the standard, non-importance sampling, Keister integrand.

Conclusion

In this work we have presented and exemplified importance sampling for (Quasi-)Monte Carlo methods. Specifically, we focused on developing the composed importance sampling framework and showing its potential to improve the efficiency of Quasi-Monte Carlo approximation. Throughout this work, the QMCPy package was used to easily setup and execute importance sampling in a flexible and intuitive framework.

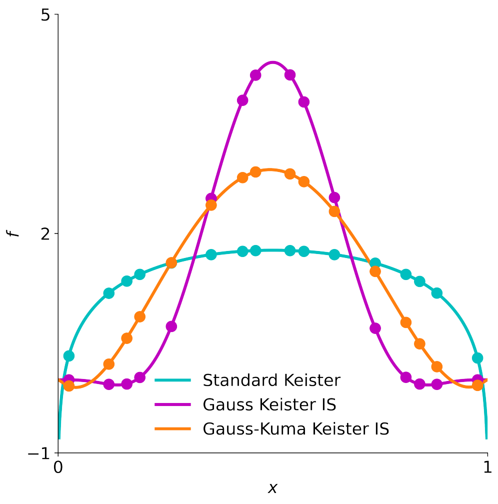

Fig. 2. The Keister integrand without importance sampling (`Standard Keister`), with Gaussian importance sampling (`Gaussian Keister IS`), and with composed Gaussian-Kumaraswamy importance sampling (`Gauss-Kuma Keister IS`). Each transformed integrand shows the sampling locations of the same 16 scrambled Sobol' points. The lower variation of importance sampling integrands makes their (Q)MC approximation more efficient.

Future Work. Control variates are another popular and powerful technique to rewrite the original integral. In the future, we plan to add support for control variates into QMCPy. We hope QMCPy's support for these performance enhancing techniques will allow (Q)MC researchers and practitioners to more easily access their benefits.

While importance sampling and control variates can provide substantial benefits to (Q)MC methods, the choice of an effective importance sampling measure or control variate function is currently a difficult and manual task. In the future we hope to develop algorithms to automatically select good importance sampling measures and control variate functionals, perhaps using machine learning. If successful, we would like to implement these methods into QMCPy to extend their benefits to a wider user base.

QMCPy Resources. To learn more about QMCPy for (Quasi-)Mote Carlo, we recommend our article for the MCQMC2020 proceedings (5) and the resources therein. Those interested in following the development of QMCPy are urged to visit qmcpy.org or view our GitHub repository at github.com/QMCSoftware/QMCSoftware.

Acknowledgements. We would like to thank SigOpt and National Science Foundation grant DMS-1522687 for their continued support of this project.

References

1. Sou-Cheng T. Choi, Fred J. Hickernell, R. Jagadeeswaran, Michael J. McCourt, and Aleksei G. Sorokin. QMCPy: A Quasi-Monte Carlo Python library, 2020+. URL https://github.com/QMCSoftware/QMCSoftware.
2. B. D. Keister. Multidimensional quadrature algorithms. *Computers in Physics*, 10:119–122, 1996. .
3. Fred J. Hickernell and Lluís Antoni Jiménez Rugama. Reliable adaptive cubature using digital sequences, 2014.
4. P. Kumaraswamy. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1):79–88, 1980. ISSN 0022-1694. . URL https://www.sciencedirect.com/science/article/pii/0022169480900360.
5. Sou-Cheng T. Choi, Fred J. Hickernell, R. Jagadeeswaran, Michael J. McCourt, and Aleksei G. Sorokin. Quasi-monte carlo software, 2021.