Referee report

Paper: Quasi-Monte Carlo Software
Authors: Choi et al.
Journal: MCQMC 2020
Date: May 2021

Overall Evaluation

This paper proposes and discusses a software tool named QMCPy, in the form of a Python library, for estimating integrals via quasi-Monte Carlo (QMC). The library offers different types of QMC point sets and randomizations. The point sets are extensible in the number of points, using successive powers of 2. The main function takes an extensible point set, an integrand, and a target maximal error (or variance), and computes an RQMC estimator by doubling the number of points successively until the error bound is below the target. Several examples are given, some with a change of measure and/or a change of variables, and the corresponding Python code.

This is certainly a very useful tool, designed by knowledgeable people, which is likely to grow and spread. The paper is reasonably well written and should have its place in the MCQMC 2020 book. However, there are several pieces that would need clarification, as well as some additional explanations to make the paper more self-contained. I understand that one can also study the tutorial and the Colab notebooks to understand how things work, but it would be good if the paper can also be understood easily just by itself.

Details

Page 3: "The measure of the difference..." should be "A measure of the difference..." (not unique).

Page 4, Eq.(5): Replace the comma by a period.

Page 4, "highly stratified sampling": A key difference between RQMC and stratification is that in the latter, once the partition is defined, the positions of the points in the different pieces are conditionally independent. Because of that, the variance can never increase compared with Monte Carlo, which is not true for RQMC. See Section 2 of [2], for example. This independence property can make it easier to prove convergence rates for the variance; see [3], Proposition 6, for an example of this.

Page 5, SSJ: You may add "in the `hups` package".

The authors call an RQMC point set a *discrete distribution*, which I find a bit confusing. The RQMC points do not really follow a discrete distribution and there is no sampling from a discrete distribution. After they are randomized, conditional on their values, the n points form

a deterministic set and one can think of a discrete uniform distribution over these n points, but one never generate points from this discrete distribution. A `DiscreteDistribution` object has a `gen_samples` method that returns all the n points of the RQMC point set in a large two-dimensional array. It is probably too late to change the name, but a comment on this could be added.

Page 6: The $\oplus$ sign is commonly used to denote the exclusive-or operation, so its use for something more general is a bit confusing.

"lattices and digital sequences prefer n to be a power of 2" should be better qualified. This is true for *digital nets in base 2*. For digital nets in prime base $b > 2$, we would prefer a power of b. For lattice rules, a prime n is sometimes better than a power of 2; see Figure 6 of [4], for example. I think the real reason for using a power of 2 for lattices in this paper is to have a sequence of embedded lattices. This should be clarified.

Bottom of table 1: the $\ominus$ operator is not defined. What is it?

From what we see in Section 4, it seems that the user cannot select the point sets (direction numbers, generating matrices, generating vectors for the lattices). A comment on this would be good.

In several places, a reader who is not already familiar with both Python and QMCPy (like me) may struggle to understand what goes on when reading the Python code. For example, on page 8, it would be good to recall quickly in the text what the parameters mean in `IIDStdUniform(2)`, `gen_samples(1)`, and `gen_samples(n_min=1,n_max=2)`.

On page 12, it would also be good to explain the code a bit. What is `CustomFun` doing? What are `Halton(2)` and `Gaussian` doing? Do we have $d = 2$ in this case?

The different ways of rewriting the integral shown on pages 12 and 13 are only remotely connected to the software itself. I suppose the software can take any reformulation as an integral over the unit hypercube. Of course, the MC and RQMC variances will depend on the reformulation. We also miss some insight about the specific choices of reformulations given here. For example, the optimal change of measure for the importance sampling is the one that makes the integrand completely flat and gives a variance of zero [1]. This zero-variance IS can often be well approximated [6]. Basically, we want λ_{IS} to be approximately proportional to $g \cdot \lambda$. But the choice made in the paper seems arbitrary and is not really justified.

The code on page 14: Say what `CubQMCSobolG` is doing exactly. How is the error or variance estimated? What randomization is used for the Sobol points?

Page 16: Why use a trapezoidal rule? I think Asian option payoffs in practice are already defined by a finite arithmetic average, no?

"`Integrand` object to approximate the Asian call option"? You mean to "approximate the value of ... "?

Bottom of page 16: How did you select the upward drift? An optimal choice that would provide zero variance would have to be dynamic, as explained in [6] for the Markov chain setting.

Section 8.1: What criteria were used to construct these lattice sequences?

Section 8.2: "better evenness for ... powers of 2" is because of the specific construction of embedded lattices. It is not always true in general. Maybe this should be clarified.

Section 8.3, first paragraph: How are the IID Gaussian points obtained if it is not by inversion?

Second paragraph: Here you consider successive changes of measures and/or successive changes of variables. It is not clear why one would want to do that. In the end, the combined changes of variables corresponds to a single change of variable, no? We miss a motivation for this.

What is the Kumaraswamy distribution?

Page 20: "Successful importance sampling places more points where the original integrand g varies more" is not quite correct. Again, see [1] and [6] for the optimal choice.

Page 22: Automatic adaptive choice of IS: How?

Reference 27 is not correct. There is the journal paper [5] on Lattice Builder. Then there is the LatNet Builder software at `https://github.com/umontreal-simul/latnetbuilder`, which has more contributors: L'Ecuyer, Munger, Marion, Godin, Puchhammer.

References

[1] S. Asmussen and P. W. Glynn. *Stochastic Simulation*. Springer-Verlag, New York, 2007.

[2] P. L'Ecuyer and E. Buist. On the interaction between stratification and control variates, with illustrations in a call center simulation. *Journal of Simulation*, 2(1):29–40, 2008.

[3] P. L'Ecuyer, C. Lécot, and B. Tuffin. A randomized quasi-Monte Carlo simulation method for Markov chains. *Operations Research*, 56(4):958–975, 2008.

[4] P. L'Ecuyer and D. Munger. On figures of merit for randomly-shifted lattice rules. In H. Woźniakowski and L. Plaskota, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2010*, pages 133–159, Berlin, 2012. Springer-Verlag.

[5] P. L'Ecuyer and D. Munger. Algorithm 958: Lattice builder: A general software tool for constructing rank-1 lattice rules. *ACM Transactions on Mathematical Software*, 42(2):Article 15, 2016.

[6] P. L'Ecuyer and B. Tuffin. Approximate zero-variance simulation. In *Proceedings of the 2008 Winter Simulation Conference*, pages 170–181. IEEE Press, 2008.