Challenges in Developing Great Quasi-Monte Carlo Software

Sou-Cheng Terrya Choi, Yuhan Ding, Fred J. Hickernell, and R. Jagadeeswaran
Aleksei G. Sorokin

Abstract Quasi-Monte Carlo (QMC) methods have developed over several decades. With the explosion in computational science, there is a need for great software that implements QMC algorithms. We summarize the QMC software that has been developed to date, propose some criteria for developing great QMC software, and suggest some steps toward achieving great software.

1 Introduction

A QMC approximation of $\mu := \mathbb{E}[f(\boldsymbol{X})]$, $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$ can be implemented in a few steps:

1. Draw a sequence of n a low discrepancy (LD) [**?**, **?**, **?**, **?**] nodes, $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ that mimic $\mathcal{U}[0,1]^d$.

Sou-Cheng Terrya Choi
Department of Applied Mathematics, Illinois Institute of Technology,
RE 220, 10 W. 32nd St., Chicago, IL 60616 e-mail: schoi32@hawk.iit.edu

Yuhan Ding
Department of Applied Mathematics, Illinois Institute of Technology,
RE 220, 10 W. 32nd St., Chicago, IL 60616 e-mail: yding2@hawk.iit.edu

Fred J. Hickernell
Center for Interdisciplinary Scientific Computation and
Department of Applied Mathematics, Illinois Institute of Technology
RE 220, 10 W. 32nd St., Chicago, IL 60616 e-mail: hickernell@iit.edu

R. Jagadeeswaran
Department of Applied Mathematics, Illinois Institute of Technology,
RE 220, 10 W. 32nd St., Chicago, IL 60616 e-mail: jrathin1@hawk.iit.edu

Aleksei G. Sorokin
Department of Applied Mathematics, Illinois Institute of Technology,
RE 220, 10 W. 32nd St., Chicago, IL 60616 e-mail: asorokin@hawk.iit.edu

2. Evaluate the integrand f at these nodes to obtain $f(\boldsymbol{x}_i)$, $i = 1, \ldots, n$.
3. Estimate the true mean, μ, by the sample mean,

$$\hat{\mu} := \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}_i). \tag{1}$$

However, the practice of QMC is often more complicated. The original problem may need to be rewritten to fit the above form and/or to facilitate a good approximation with small n. Practitioners may wish to determine n adaptively to satisfy a prescribed error tolerance.

In the next section, we describe why QMC software is important. We then discuss the characteristics of great QMC software:

- Integrated with related libraries (Sect. **??**),
- Correct (Sect. **??**),
- Efficient in computational time and memory (Sect. **??**),
- Accessible to practitioners and theorists alike (Sect. **??**), and
- Sustainable by a community that owns it (Sect. **??**).

In each section we describe the challenges faced and how they might be overcome.

We draw on our collective experiences as members of the academic QMC community in the, developers of open-source and commercial scientific software, and users of a wide range of scientific software as research scientists or data scientists. Many of the insights that we have gained have come through our development of the Guaranteeed Automatic Integration Library (GAIL) [**?**] in MATLAB and the Quasi-Monte Carlo Python library (QMCPy) [**?**]. Fred: Sou-Cheng, not sure what this means. In particular, QMCPy is not only a QMC software, but also a scientific middleware framework, which provides access to more than X (Q)MC scientific software such as A [], B [], C [].

2 Why Develop QMC Software

Recent interest in quality scientific software is exemplified by the 2021 US Department of Energy *Workshop on the Science of Scientific-Software Development and Use* [**?**,**?**]. This workshop not only discussed diagnostics and treatments for the challenges of developing great scientific software, but also emphasized the importance of implementing theoretical advancements into well written, accessible software libraries. Three cross-cutting themes arose in this workshop:

- We need to consider both human and technical elements to better understand how to improve the development and use of scientific software.
- We need to address urgent challenges in workforce recruitment and retention in the computing sciences with growth through expanded diversity, stable career paths, and the creation of a community and culture that attract and retain new generations of scientists.

- Scientific software has become essential to all areas of science and technology, creating opportunities for expanded partnerships, collaboration, and impact.

These themes apply to QMC software in particular, as well as scientific software in general.

2.1 QMC Theory to Software

QMC software makes theoretical advances in QMC available to practitioners. However, translating pseudo-code into good executable code is nuanced. Software developers must write code that is computationally efficient, numerically stable, and provides reasonable default choices of tuning parameters. Great QMC software relieves users of these concerns.

2.2 QMC Software to Theory

Great QMC software opens up new application areas for QMC methods by allowing practitioners to compare new methods to existing ones. QMC software has been successful in quantitative finance, uncertainty quantification, and image rendering. Unexpectedly good or bad computational results lead to open theoretical questions. For example, the early application of QMC to high dimensional integrals arising in financial risk [**?**] led to a wave of theoretical results on the tractability of integration in weighted spaces [**?**, **?**, **?**].

Great QMC software in eliminates the need researchers to resort to "do-it-yourself" for established algorithms. The less code we have to write, the fewer errors.

We next discuss the characteristics of great QMC software outlined in the introduction. For each characteristic, we identify what is lacking and what might be done to remedy the lack.

3 Integrated

Expanding on the problem formulation in the introduction, we want to approximate well the integral or expectation, μ, by the sample mean, $\hat{\mu}$:

$$\mu := \int_{\mathcal{T}} g(\boldsymbol{t})\, \lambda(\boldsymbol{t})\, \mathrm{d}\boldsymbol{t} = \mathbb{E}[f(\boldsymbol{X})] = \int_{[0,1]^d} f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \approx \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}_i) =: \hat{\mu}_n, \quad (2)$$

$$|\mu - \hat{\mu}_n| \le \varepsilon. \quad (3)$$

Progressing from the original problem to a satisfactory solution requires several software components, which we describe in the subsections below. Great QMC software environments allow different implementations of these components to be freely interchanged.

3.1 LD Sequence Generators

A lattice, digital sequence, or Halton sequence generator typically supplies the sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$. These sequences may be deterministic or random and are intended to have an empirical distribution that approximates well the uniform distribution.

LD generators are the most prevalent generally available components of QMC software. These generators appear in Association of Computing Machinery publications [**?**, **?**, **?**], FinDer [**?**, **?**], libseq [**?**, **?**], BRODA [**?**], NAG [**?**], MATLAB [**?**], SamplePack [**?**], Grünschloß's website [**?**], the Magic Point Shop [**?**], R [**?**], Julia [**?**], PyTorch [**?**], SAS [**?**], SciPy [**?**], and TensorFlow [**?**], MatBuilder [**?**], and QMCPy [**?**].

Lattice and digital sequences are not unique, and improved or alternate versions continue to be constructed. Stephen Joe and Frances Kuo [**?**, **?**, **?**] have proposed Sobol' generator direction numbers, which are now widely used, but were not part of early LD sequence software. Pierre L'Ecuyer and collaborators have created LatNetBuilder [**?**] to construct good lattice and polynomial lattice sequences based on user-defined criteria. During a lunch at MCQMC 2022, it was proposed that we agree on a consistent format for storing the parameters that define good LD sequences so that they can be shared across software libraries.

Users computing solutions in comprehensive software environments, such as NAG, MATLAB, R, SciPy, and TensorFlow, have convenient access to LD sequence generators. However, most of these implementations in large libraries could provide more flexible LD offerings. Some large environments, such as Dakota [**?**] and SAS, have quite limited LD generators and should expand their offerings. Developers of smaller libraries with more QMC features, like BRODA and QMCPy, should demonstrate how their libraries can connect well with other software environments. An example of QMCPy connecting with other software is given in the next subsection.

3.2 Integrands and Variable Transformations

The original integral or expectation arising from the application is defined in terms of the integrand, g, and the non-negative weight, λ. For example,

- g is the discounted payoff of a financial derivative and λ is the probability density function (PDF) for a discretized Brownian motion [**?**],
- g is the velocity of a fluid at a point in rock with a random porosity field whose discretized PDF is λ [**?**], or