The Journal of Open Source Software

Quasi-Monte Carlo Methods in Python

Pamphile T. Roy [1¶], **Art B. Owen** [2], **Maximilian Balandat** [3], **and Matt Haberland** [4]

1 Quansight **2** Stanford University **3** Meta **4** California Polytechnic State University, San Luis Obispo, USA ¶ Corresponding author

DOI: 10.21105/joss.05309

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @yangyushi
- @ptmerz

Submitted: 17 February 2023
Published: 23 April 2023

Summary

NumPy random number generators and SciPy distributions are widely used to generate random numbers. However, challenges might arise when sampling in high dimensions. Quasi-Monte Carlo (QMC) methods provide an answer to these problems but are arguably hard to use. Thanks to new developments in SciPy, a new submodule was introduced in version 1.7.0 making state-of-the-art QMC methods available: `scipy.stats.qmc`.

Statement of need

NumPy pseudorandom number generators (`numpy.random`) have become the de facto standard for sampling random numbers in the scientific Python ecosystem. These methods are fast and reliable, and the results are repeatable when a seed is provided. However, sampling in high dimensions with pseudorandom numbers tends to produce gaps and clusters of points. When these random numbers are used in algorithms (including sampling, numerical integration, optimization) to solve deterministic problems, the resulting "Monte Carlo" (MC) methods have a low convergence rate. In practice, this can mean that substantial computational resources are required to provide sufficient accuracy.

In Quasi-Monte Carlo (QMC) methods (Niederreiter, 1992), the random numbers of Monte Carlo methods are replaced with a deterministic sequence of numbers that possesses many of the characteristics of a random sequence (e.g. reduction of variance with increasing sample size), but without these gaps and clusters. QMC determinism is independent of its implementation, language, and platform – the sequence is mathematically defined.

In many cases, a QMC sequence can be used as a drop-in replacement for a random number sequence, yet they are proven to provide faster convergence rates (both in theory and practice) (Owen, 2019). When true stochasticity is required (e.g. statistical inference), QMC sequences can be "scrambled" using random numbers, and several smaller scrambled QMC sequences can often replace one large random sequence.

QMC methods were added to SciPy (Virtanen et al., 2020) after an extensive review and discussion period (Roy et al., 2021) that lead to a very fruitful collaboration between SciPy's maintainers and renowned researchers in the field. For instance, our implementation inspired additional work on the importance of including the first point in the Sobol' sequence (Owen, 2020).

The following set of QMC features are now available in SciPy:

- Sobol' and Halton sequences (scrambled and unscrambled),
- Poisson disk sampling,
- Quasi-random multinomial and multivariate normal sampling,
- Discrepancy measures (C^2, wrap around, star-L_2, mixed),

- Latin Hypercube Sampling (centered, strength 1 or 2),
- Optimize a sample by minimizing C^2 discrepancy or performing Lloyd-Max iterations,
- Fast numerical inverse methods to sample arbitrary univariate distributions with QMC (Baumgarten & Patel, 2022),
- QMC integration.

Before the release of SciPy 1.7.0, the need for these functions was partially met in the scientific Python ecosystem by tutorials (e.g. blog posts) and niche packages, but the functions in SciPy have several advantages:

- Popularity: With millions of downloads per month, SciPy is one of the most downloaded scientific Python packages. New features immediately reach a wide range of users from all fields.
- Performance: The low-level functions are written in compiled languages such as Cython and optimized for speed and efficiency.
- Consistency: The APIs comply with the high standards of SciPy, function API reference and tutorials are thorough, and the interfaces share common features complementing other SciPy functions.
- Quality: As with all SciPy code, these functions were rigorously peer-reviewed for code quality and are extensively unit-tested. In addition, the implementations were produced in collaboration with the foremost experts in the QMC field.

Since the first release of all these new features, we have seen other libraries add support for and rely on SciPy's implementations, e.g. Optuna (Ishikawa et al., 2022) and SALib (Roy & Iwanaga, 2022).

Acknowledgements

The authors thank professors Sergei Kucherenko (Imperial College London) and Fred Hickernell (Illinois Institute of Technology) for helpful discussions. The SciPy maintainer team provided support and help regarding the design and integration, notably Ralf Gommers (Quansight) and Tyler J. Reddy (Los Alamos National Laboratory).

References

Baumgarten, Christoph, & Patel, Tirth. (2022). Automatic random variate generation in Python. In Meghann Agarwal, Chris Calloway, Dillon Niederhut, & David Shupe (Eds.), *Proceedings of the 21st Python in Science Conference* (pp. 46–51). https://doi.org/10.25080/majora-212e5952-007

Ishikawa, K., Imamura, H., & Roy, P. T. (2022). Add QMC sampler. In *GitHub pull request*. GitHub. https://github.com/optuna/optuna/pull/2423

Niederreiter, H. (1992). *Random number generation and Quasi-Monte Carlo methods*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611970081

Owen, A. B. (2019). *Monte Carlo book: The Quasi-Monte Carlo parts*. https://artowen.su.domains/mc/

Owen, A. B. (2020). On dropping the first Sobol' point. *arXiv e-Prints*. https://doi.org/10.1007/978-3-030-98319-2_4

Roy, P. T., Balandat, M., Owen, A. B., & Haberland, M. (2021). ENH: Add stats.qmc module with quasi Monte Carlo functionality. In *GitHub pull request*. GitHub. https://github.com/scipy/scipy/pull/10844

Roy, P. T., & Iwanaga, T. (2022). Add Sobol' sampler from SciPy. In *GitHub pull request*. GitHub. https://github.com/SALib/SALib/pull/519

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., & Bright, J. et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2