Eurographics Symposium on Rendering (DL-only Track) (2021)
A. Bousseau and M. McGuire (Editors)

Stochastic Generation of (t,s) Sample Sequences: Supplemental Materials

Andrew Helmer Per Christensen[1] Andrew Kensler[2]

[1]Pixar Animation Studios [2]Amazon

Abstract

Supplemental material for "Stochastic Generation of (t,s) Sample Sequences". Discussion of pmj02 generator matrices. Optimizations for generating (0,2)-sequences. Comparison of higher dimensions of Halton sequences with different scrambling methods. Render comparisons with different sequences.

1. Generating pmj02 sequences

Christensen et al. [CKK18] described an algorithm for stochastic generation of (0,2)-sequences, known as pmj02 sequences. Their paper gave a specific ordering of the sequences: after generating 2^m points, the sequence is extended to 2^{m+1} points by iterating over the previous points and generating new points in diagonally opposite subquadrants of the unit square.

This means that all disjoint subsequences of 4^m points will iterate over the $2^m \times 2^m$ grid in the exact same order. From a generator matrix perspective, the higher digits in the sample index must not permute the lower digits. Specifically, the j'th column of the i'th row must be zero for all $j > 2i$. This property makes it a *finite-row* matrix, a term coined by Hofer and Larcher [HL10].

1.1. Construction of finite-row generator matrices

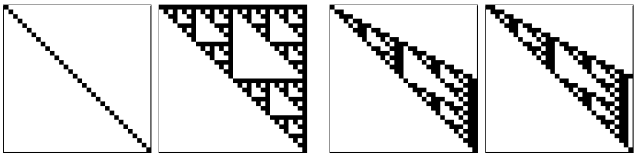
It is possible to perform a Faure-Tezuka scrambling to convert the Faure sequence matrices to finite-row matrices. Hofer and Pirsic [HP11] give a construction using an upper triangular matrix of Stirling numbers of the first kind. We present a slightly simplified version from their paper. The scrambling matrix $\mathbf{S}$ is given as:

$$\mathbf{S} = \begin{pmatrix} \begin{bmatrix}0\\0\end{bmatrix} & \begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}2\\0\end{bmatrix} & \cdots \\ 0 & \begin{bmatrix}1\\1\end{bmatrix} & \begin{bmatrix}2\\1\end{bmatrix} & \cdots \\ 0 & 0 & \begin{bmatrix}2\\2\end{bmatrix} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \mod b,$$

where $\begin{bmatrix}n\\k\end{bmatrix}$ is the unsigned Stirling number of the first kind, calculated with the recurrence:

$$\begin{bmatrix}n\\k\end{bmatrix} = \begin{cases} 1 & \text{if } n = k = 0 \\ 0 & \text{if } (k=0) \neq (n=0) \\ \begin{bmatrix}n-1\\k-1\end{bmatrix} + (n-1)\begin{bmatrix}n-1\\k\end{bmatrix} & \text{otherwise.} \end{cases}$$

For a base-b Faure $(0,b)$ sequence, multiplying each of the Pascal matrices with this scrambling matrix gives a set of b finite row generator matrices $\{\mathbf{P}^0\mathbf{S}, \mathbf{P}^1\mathbf{S}, \ldots, \mathbf{P}^{b-1}\mathbf{S}\}$, where the j'th column of the i'th row is zero for all $j > bi$. Figure 1 shows a comparison between the Sobol (0,2)-sequence generator matrices, and the finite-row matrices that can be used to generate the pmj02 sequence.



(a) *Sobol (0,2) matrices* **(b)** *Finite-row (0,2) matrices*

Figure 1: *Generator matrices of the Sobol (0,2)-sequence compared to finite-row matrices used to generate the pmj02 sequence.*

These matrices can again be used to calculate a set of xor-values, which are given in the supplemental code. We can compare these xor-values to ones which are taken from a pregenerated pmj02 sequence, as long as the pmj02 sequence has optimal disjoint subsequences. When extending the sequence from an odd to an even power-of-two, there is a question of which subquadrant to generate the next point in. Brown [Bro19] observed that optimal subsequences could be obtained implicity by using the subquadrant opposite to the initial subquadrant on the x-axis. For instance, if the

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.

first point was placed in the $[0, 0.5)$ interval on the x-axis, and the second in the interval $[0.5, 1)$, the third would also be placed in the $[0.5, 1)$ interval (opposite the first point).

Using sequences generated with Brown's technique, we retrospectively calculated xor-values by finding earlier points that share the relevant interval with a later point. These xor-values were identical to values derived from the analytic construction, only with the x-dimension and y-dimension switched. By instead using the subquadrant opposite on the y-axis, they were identical.

1.2. Properties of finite-row sequences

Finite-row sequences have interesting properties that may be useful in graphics. The Stirling number construction guarantees a b-dimensional jittering for *all* subsequences, not only disjoint ones. For example, any subsequence of 4^m pmj02 points will be base-2 jittered in 2D, i.e. stratified with the $2^m \times 2^m$ *grid*.

This property allows for a particular generalization of Halton sequences. Coprime base finite-row $(0, b)$-sequences can be juxtaposed, and the resulting sequence will have high-dimensional mixed-base stratification. For example, a finite-row base-2 $(0, 2)$-sequence can be combined with a finite-row base-3 $(0, 3)$-sequence, and any subsequence of $4^m 27^n$ points will be stratified in the $2^m \times 2^m \times 3^n \times 3^n \times 3^n$ grid. Hofer and Pirsic [HP13] referred to these as "finite-row Faure-Halton" sequences. Van der Corput sequences can also be used, e.g. the first (base-2) dimension of the Halton sequence can be replaced with a 2D pmj02 sequence, and the resulting sequence still achieves guaranteed multidimensional stratification.

2. Optimizations for base-2 performance comparisons

Table 2 of the paper compares performance of different methods of generating $(0, 2)$-sequences. To make the comparison as fair as possible, we applied some straightforward optimizations to some of the methods, which we outline here. Every change detailed here was tested to verify that it improved performance.

Hash-based scrambling. We used the code provided in the supplemental materials of Burley [Bur20]. For a fair comparison, we omitted the index shuffling of the sequence. The generator matrix multiplication was removed for the first dimension (as it's the identity matrix), and the redundant bit reversal was then omitted. We also added an early exit in the matrix multiplication, when all remaining digits of the index are zero. The net of these changes was approximately a 2x improvement in performance.

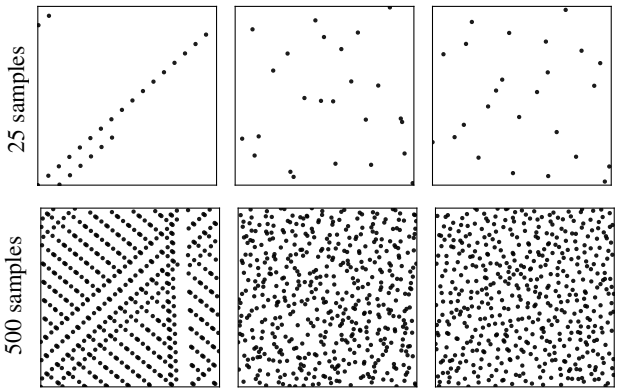
Lazy permutation trees. We omit the initialization of the `RandomizedTSSequence_base_2` object, as suggested by Friedel and Keller [FK02]. Our timings only include the subsequent calls to `::random_restart()` and the iteration to obtain each coordinate.

Efficient generation. Pharr [Pha19] traverses the strata and constructs a tree representing available strata for each point, then traverses the tree to get the valid 1D strata. We omit the tree construction, and store valid 1D strata in the initial traversal. This improves performance ~2.5x, both locally and in the comparison between our numbers and the numbers provided by Pharr.

Stochastic generation. Our implementation is sensitive to the calls to the pseudorandom number generator. Using either the C++ standard library LCG, or the operating system's `drand48()`, we achieve roughly 50M samples/second. By redefining `drand48()`'s implementation in a header file, that performance jumps to 215M samples/second. Using a PCG32 pseudorandom number generator, also defined in a header file, gives 195M samples/second. This may be due to inlining the random number generation, as performance does not improve if the PRNG is only declared in the header file but not implemented. Using a table of precomputed uniform random numbers, stochastic generation performance improves further to 246M samples/sec.

3. Comparison of higher dimensional Halton sequences

Section 5.1 showed a comparison of the Halton 2,3 sequence with no scrambling, random stochastic generation, and best-candidate samples. Here Figure 2 shows the same comparison, but with the Halton 17,19 sequence.



(a) *Unscrambled* **(b)** *Stochastic* **(c)** *Best-candidate*

Figure 2: *Comparison of Halton 17,19 sequences generated with different scrambling methods. Stochastic generation breaks apart correlations and best-candidate samples improve the distances between points.*

Higher bases are considerably more in need of scrambling. Additionally, higher bases benefit more from best-candidate samples, when candidates can be generated from all available strata offsets.

4. Render comparisons

Figure 3 compares path tracing errors using sequences generated with our method against sequences shipped with PBRT [PJH17]. Out of the many possible sequences, we choose five to evaluate:

*Owen-scrambled Sobol' (0,2) (**ssobol02**).* 2D and 1D Sequences are generated with stochastic generation and shuffled to decorrelate higher dimensions ("padding"). Virtually identical in error to the pmj02 sequences of Christensen et al. [CKK18].

*Stochastic Sobol' (0,2) with best-candidates (**ssobol02bn**).* The same as ssobol02, but each new sample point is chosen from the

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.

best of 100 candidates, maximizing the minimum distance from previous points.

*High-dimensional Owen-scrambled Sobol' (**ssobol**).* Each pixel uses an Owen-scrambled Sobol' sequence, up to the full dimensionality of the render.

*Stochastic Faure (0,5) (**sfaure05**).* PBRT allows the renderer to request 1D or 2D samples. These requests are assigned greedily to (0,5)-sequences, stochastically generated with correlated swapping, padded for higher dimensions. Dimensions are skipped to prevent a 2D sample from splitting (0,5)-sequences.

*Stochastic Halton (**shalton**).* Each pixel uses a stochastically generated Halton sequence, up to the full dimensionality of the render. Samples are generated using correlated swapping.

All eight sample sequences have good performance and each can be considered state-of-the-art. The Halton sequences perform worse than the sfaure05 sequence in all full renders, but not all crops, and Halton sequences are more flexible with sample count, while the sfaure05 only performs optimally at powers of five. Out of the non-Halton sequences, no two sequences could be compared where one is better than the other in all five renders.

References

[Bit16] BITTERLI, B. *Rendering Resources*. https://benedikt-bitterli.me/-resources/. 2016 4.

[Bro19] BROWN, S. *Progressive Multi-Jittered Sample Sequences*. https://github.com/sjb3d/pmj. 2019 1.

[Bur20] BURLEY, B. "Practical hash-based Owen scrambling". *Journal of Computer Graphics Techniques* 10.4 (2020), 1–20 2.

[CKK18] CHRISTENSEN, P., KENSLER, A., and KILPATRICK, C. "Progressive multi-jittered sample sequences". *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* 37.4 (2018), 21–33 1, 2.

[FK02] FRIEDEL, I. and KELLER, A. "Fast generation of randomized low-discrepancy point sets". *Monte Carlo and Quasi-Monte Carlo Methods 2000*. Ed. by FANG, K.-T., NIEDERREITER, H., and HICKERNELL, F. Springer, 2002, 257–273 2.

[HL10] HOFER, R. and LARCHER, G. "On existence and discrepancy of certain digital Niederreiter-Halton sequences". *Acta Arithmetica* 141 (2010), 369–394 1.

[HP11] HOFER, R. and PIRSIC, G. "An explicit construction of finite-row digital (0,s)-sequences". *Uniform Distribution Theory* 6.2 (2011), 13–30 1.

[HP13] HOFER, R. and PIRSIC, G. "A finite-row scrambling of Niederreiter sequences". *Monte Carlo and Quasi-Monte Carlo Methods 2012*. Ed. by DICK, J., KUO, F., PETERS, G., and SLOAN, I. Springer, 2013, 427–437 2.

[Pha19] PHARR, M. "Efficient generation of points that satisfy two-dimensional elementary intervals". *Journal of Computer Graphics Techniques* 8.1 (2019), 56–68 2.

[PJH17] PHARR, M., JACOB, W., and HUMPHREYS, G. *Physically Based Rendering: From Theory To Implementation*. 3rd. Morgan Kaufmann, 2017 2, 4.

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.

Figure 3: *Root mean square errors of images rendered with path tracing, comparing sequences generated with our new stochastic generation (left five columns) and existing sequences in PBRT [PJH17] (right three columns). Lowest errors are in bold text. Each sequence is competitive in different circumstances. With the capability of efficiently generating many sequences, more research is needed into choosing the best sequences for a given render and into optimally assigning integration dimensions.*

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.