MatBuilder: Mastering Sampling Uniformity Over Projections
Lois Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Alexander Keller, Victor Ostromoukhov

To cite this version:
Lois Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Alexander Keller, et al.. MatBuilder: Mastering Sampling Uniformity Over Projections. ACM Transactions on Graphics, Association for Computing Machinery, 2022, 41 (4), pp.84. 10.1145/3528223.3530063. hal-03664112

HAL Id: hal-03664112
https://hal.archives-ouvertes.fr/hal-03664112
Submitted on 23 May 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
MatBuilder: Mastering Sampling Uniformity Over Projections

LOIS PAULIN, Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, France
NICOLAS BONNEEEL, Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, France
DAVID COEURJOLLY, Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, France
JEAN-CLAUDE IEHL, Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, France
ALEXANDER KELLER, NVIDIA, Germany
VICTOR OSTROMOUKHOV, Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, France

Many applications ranging from quasi-Monte Carlo integration over optimal control to neural networks benefit from high-dimensional, highly uniform samples. In the case of computer graphics, and more particularly in rendering, despite the need for uniformity, several sub-problems expose a low-dimensional structure. In this context, mastering sampling uniformity over projections while preserving high-dimensional uniformity has been intrinsically challenging. This difficulty may explain the relatively small number of mathematical constructions for such samplers. We propose a novel approach by showing that uniformity constraints can be expressed as an integer linear program that results in a sampler with the desired properties. As it turns out, complex constraints are easy to describe by means of stratification and sequence properties of digital nets. Formalized using generator matrix determinants, our new MatBuilder software solves the set of constraints by iterating the linear integer program solver in a greedy fashion to compute a problem-specific set of generator matrices that can be used as a drop-in replacement in the popular digital net samplers. The samplers created by MatBuilder achieve the uniformity of classic low discrepancy sequences. More importantly, we demonstrate the benefit of the unprecedented versatility of our constraint approach with respect to low-dimensional problem structure for several applications.

ACM Reference Format:

1 INTRODUCTION

Generating samples covering a domain as uniformly as possible is core to many scientific domains including computer graphics. It is well known that simple Monte Carlo integration converges faster when using well-spaced samples than using random sampling, which is particularly useful in physically based rendering [Pharr et al. 2016]. Stratified sampling of a population improves the precision of statistical surveys [Kish 1965]. Generating highly uniform samples is also important in Deep Learning [Keller and Van keirsbilck 2022] and stratification can be used to prevent class imbalance during learning [Yuan et al. 2018]. Qualifying and quantifying this uniformity is a challenge in itself and has been the scope of decades of research, involving tools such as discrepancy [Hickernell 1998] and optimal transport [Paulin et al. 2020]. Applications may require uniformity, not only in the possibly high-dimensional space of samples, but also for particular lower-dimensional projections. This is notably the case for rendering [Perrier et al. 2018], where integration is performed over a path space consisting of the union of domains of increasing dimensions as light bounces – uniformity is then desired for each integration subdomain. While generating a specific set of samples satisfying complex combinations of constraints may be performed via energy optimization, certain applications require the ability to generate sequences of points instead of a single point set, that is, the ability of adding more samples without touching the existing ones, while preserving uniformity constraints. For instance, rendering applications may progressively show the rendered image being refined or adaptively terminate rendering based on image quality. In addition, the efficiency of sample generation is crucial and it may not be acceptable for point set optimization time to exceed the Monte Carlo simulation itself.

Such constraints are dictated by the user and are application specific. While low-discrepancy sequences have been designed and used as general-purpose samplers – this is the case of the popular family of Sobol’ sequences – they may not be adapted to the user’s requirements.

In this paper, we introduce the mathematical tools and a description language that allow one to define sample point sets and sequences that satisfy complex uniformity constraints. We achieve this feat with two key ideas. First, we show that these constraints can be written in the form of constraints on matrix determinants. This formalization allows us to represent the constraints as an integer linear program [Wolsey 2020]. Second, we show that higher-order finite fields offer many more degrees of freedom to satisfy constraints, overcoming restrictions of classic algebraic constructions. Solving the integer linear program lets us create application-specific samplers. We demonstrate the superior performance of the resulting samplers for rendering, texture exploration, and optimal control applications. The source code of the proposed MatBuilder system has been released to facilitate further research [Paulin et al. 2022].
2 RELATED WORK

In numerical integration, quasi-Monte Carlo methods can outperform Monte Carlo methods by using correlated samples that are more uniformly distributed than independent uniformly distributed random samples. Quasi-Monte Carlo methods have profoundly impacted computer graphics and are now standard in rendering technologies [Pharr et al. 2016]. We will review the work most relevant to our new construction.

Low-discrepancy sequences. Deterministic number-theoretic constructions of highly uniform point sets and sequences form the foundation of quasi-Monte Carlo methods [Halton 1964; Niederreiter 1992; Sobol’ 1967]. Their uniformity is measured by discrepancy [Hickernell 1998] that is defined as the supremum of the absolute difference between the relative number of samples that fall into an arbitrary convex subset of the unit cube and the volume of the subset. Point sequences that achieve a discrepancy of order $O(\log N/N)$ are called low-discrepancy sequences. The classic theory and constructions of low discrepancy sequences are extensively covered in reference textbooks by Niederreiter [1992], Lemieux [2009], and Dick and Pillichshammer [2010].

Our contribution draws from the algorithmic framework for the generation of low-discrepancy sequences and hence can be a drop-in replacement for the improvement of existing codes.

Projective samplers. Already Sobol’ [1967] recognized that there are combinatorial lower bounds on the achievable uniformity of low-dimensional projections of his algebraic high-dimensional construction. Later Sobol’ et al. [2011] introduced the concept of constraints to improve the uniformity of low-dimensional projections at the cost of relaxed high-dimensional uniformity. Similarly, improvements to the low-dimensional projections of the Sobol’ sequence have been identified by computer search [Joe and Kuo 2008]. More recently, computer search has been key to improving the quality parameter $t$ of a generalization of Sobol’ construction [Faure and Lemieux 2016, 2019].

Recognizing the particular structure of physically-based rendering integration domains, for example obtained by repeated scattering in path tracing algorithms, computer graphics has been at the forefront of improving the uniformity of point sets in low dimensional projections using both computer search [Ahmed et al. 2016; Marques et al. 2020; Paulin et al. 2020; Perrier et al. 2018] and mathematical constructions [Paulin et al. 2021].

As compared to the aforementioned specific approaches, we come up with a general formulation of constraints and a system to solve them in an efficient way.

Orthogonal Arrays. Orthogonal Arrays (OA) are widely used in engineering and the statistical design of scientific experiments [Bose and Bush 1952; Bush 1952; Plackett and Burman 1946]. Given a set of values in $\{0, \ldots, b - 1\}$, an OA in dimension $s$ of strength $t \leq s$ is set of $n$ points seen as an array $A$ of size $n \times s$ where each submatrix of size $n \times t$ (i.e. $t$ dimensional projection) contains all $b^t$ possible distinct rows [Owen 2013].

As introduced to computer graphics [Jaroz et al. 2019], OAs generalize the notion of Latin hypercube and (multi-)jittered sampling, focusing on stratification properties on some $t$--dimensional projections of the samples. So-called strong OAs fill the gap between orthogonal arrays and low discrepancy nets [He and Tang 2013].

The constraints implied by orthogonal arrays are a special case of our general approach to designing generator matrices.

Randomization. Low-discrepancy sequences may be randomized to allow for variance estimation and unbiased Monte Carlo integration [Owen 1998]. The underlying principles are recursive random scrambling and random shifting on the unit torus. These techniques may also be used for the optimization of sampling points according to various criteria: besides minimizing discrepancy, the mutual minimum distance of the points may be maximized and the spectral properties may be tweaked to approximate blue noise characteristics in low dimensions [Ahmed et al. 2016; Ahmed and Wonka 2020, 2021; Heitz et al. 2019].

While many improvements of scrambled algebraic samplers are intrinsic to our constraint-based approach, our samplers are compatible with the aforementioned randomization techniques.

3 PRELIMINARIES

Our new method is based on the classical algorithms (reviewed in Section 3.1) to generate digital nets and sequences [Dick and Pillichshammer 2010; Lemieux 2009; Niederreiter 1992] in the $s$-dimensional unit cube $[0, 1)^s$, where a set of generator matrices $C_0, \ldots, C_{s-1} \in \mathbb{F}_p^{m \times m}$ defines an $s$-dimensional point set of size $p^m$. Rather than constructing such matrices using number theory, we aim to design the generator matrices $C_0, \ldots, C_{s-1}$ by specifying constraints that we introduce in Section 4 as design principles derived from the algebraic properties of the classic constructions (see Section 3.2).

### 3.1 Digital Nets and Sequences from Generator Matrices

To compute the $i$-th component $x_{a,i}$ of the $a$-th point $x_a$, the index (or its ordinal number) $a \in \mathbb{N}_0$ is represented in base $p$ as the vector $(a_0, \ldots, a_{m-1})$ of $m$ digits, where $a_0$ is the least-significant digit. Given the $i$-th generator matrix $C_i$, the vector

$$\begin{pmatrix} b_{m-1} \\ \vdots \\ b_0 \end{pmatrix} := C_i \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{m-1} \end{pmatrix}$$

is mapped to the unit interval by

$$x_{a,i} := \frac{1}{p^m} \sum_{j=0}^{m-1} b_j \cdot p^j \in [0, 1).$$

The matrix multiplication is performed in the finite (Galois) field $\mathbb{F}_p$ (sometimes denoted GF(p)). Using a finite field with a base $p$ that is a prime power rather than a prime would involve mapping the digits $a_k$ to the finite field and mapping the digits $b_k$ back into the integers [Niederreiter 1992, Section 4.3]. Applying only prime bases in this article, these mappings are not required and operations are simply performed modulo $p$.

The above algorithm has become extremely popular, as in base $p = 2$, the vector operations on $\mathbb{F}_2^{m \times m}$ can be efficiently implemented as bit-vector ‘xor‘ and ‘and‘ operations for addition and multiplication,
Fig. 1. **Generator matrix structure:** (a) The first row of the generator matrix $C_i$ acts on the most significant digit of $b_{m-1}$, selecting one two intervals of size $2^{m-1}$ in base 2, the first two rows of $C_i$ determine one intervals of size $2^{m-2}$ and so on. (b) When coupling the matrices to design a 2D sampler, the first rows of $C_0$ select intervals along the $x$-axis, while the first rows of the $C_1$ select intervals along the $y$-dimension. If $M_K$ is of full rank, samples generated by $C_0$ and $C_1$ will be stratified in the cells induced by selected rows of the two matrices.

respectively. For higher bases, efficient implementations process multiple digits simultaneously using lookup tables [Keller et al. 2012].

### 3.2 Stratification of Digital Nets and Sequences

Digital nets and sequences generated by matrices imply a stratification of the unit cube as illustrated in Figure 3. These strata are specified by

Definition 3.1 (see [Niederreiter 1992, p.48]). For a fixed dimension $s \geq 1$ and an integer base $b \geq 2$ the subinterval

$$E = \prod_{j=1}^{s} [a_j, b_{j+1}] = \prod_{j=1}^{s} [a_j \cdot b^{-d_j}, (a_j + 1) \cdot b^{-d_j}] \subseteq \mathbb{P}$$

with $0 < a_j < b^{d_j}$, $a_j, d_j \in \mathbb{N}_0$, is called an **elementary interval** in base $b$.

The stratification and hence uniformity constraints given by all the elementary intervals can be used to characterize the uniformity of point sets and sequences:

Definition 3.2 (see [Niederreiter 1992, Def. 4.1]). For integers $0 \leq t \leq m$, a $(t, m, s)$-net in base $b$ is a point set of $b^m$ points in $[0, 1]^s$ such that there are exactly $b^t$ points in each $b$-adic elementary interval $E$ with volume $b^{t-m}$.

Definition 3.3 (see [Niederreiter 1992, Def. 4.2]). For an integer $t \geq 0$, a sequence $x_0, x_1, \ldots$ of points in $[0, 1]^s$ is a $(t, s)$-sequence in base $b$ if, for all integers $k \geq 0$ and $m > t$, the point set $x_{k+b^m}, \ldots, x_{k+b^{m-1}}$ is a $(t, m, s)$-net in base $b$.

Revisiting the point set in Figure 3, we see a realization of a $(0, 4, 2)$-net in base $b = 2$ that is an $s = 2$-dimensional point set with $2^4 = 16$ points, where $t = 0$, i.e. exactly one point is in each possible elementary interval. A smaller quality parameter $t$ implies a lower discrepancy and hence better uniformity of a point set or sequence.

Historically, Sobol’ construction [Sobol’ 1967] has led to the above definitions and in fact, the Sobol’ low discrepancy sequence is a $(t, s)$-sequence in base 2. Its generator matrices $C_i$ are constructed from the $i$-th primitive polynomial over $\mathbb{F}_2$, Joe and Kuo [2008] have published Sobol’ generator matrices with optimized low-dimensional projections up to dimension $s = 21001$ that are widely used in computer graphics and numerical simulation in general. Grünschloß et al. [2008] have performed an exhaustive search on the binary matrix $C_1$ such that starting from the identity matrix $C_0 = I$, the point sets generated by $(C_0, C_1)$ are $(0, m, 2)$-nets maximizing the minimal distance between the samples. More recently, Paulin et al. [2021] have proposed a cascaded construction of the $C_i$ matrices in base 2 in a way that any pair of consecutive matrices $(C_i, C_{i+1})$ defines a $(0, m, 2)$-net in base 2, with optimized discrepancy on the remaining two-dimensional projections. Ahmed and Wonka [2021] further explore all possible dyadic nets in base 2 but only for 2 dimensions. Still in base 2, L’Ecuyer and Munger [2016] propose a generic tool, LatNetBuilder, for constructing matrices in higher dimensions optimizing a given figure of merit (e.g. full-space or projective discrepancy), using either an exhaustive or a randomized exploration.

There remain severe limitations to a low base. For example, achieving the best quality parameter $t = 0$ is impossible for $s > p$. While there exist constructions with $t = 0$ in higher bases, a higher base results in a higher discrepancy and hence reduced uniformity. In general, it is considered hard to construct algebraic low-discrepancy sequences. Since we aim to improve the uniformity of low-dimensional projections, too, the problem becomes even harder. Therefore, we propose a novel, constraint-based approach to digital nets and sequences.

### 4 GENERATOR MATRIX CONSTRAINTS

In this section we show how classic uniformity constraints translate into determinant inequalities that can be used as design constraints.
4.1 How Generator Matrices Act

The algorithm of Equation (1) is illustrated in Figure 1: the j-th row of a generator matrix \( C_1 \) determines the j-th digit \( b_{m-j} \). The first \( k \) rows of \( C_1 \) determine the \( k \) most significant digits of the vector \( b \). They in turn determine in which of the \( p^k \) uniformly sized intervals \( [1 \cdot p^{m-k}, (l+1) \cdot p^{m-k}) \), \( 0 \leq l < p^k \) the sample will be placed.

If the generator matrix is of full rank, that is its determinant is non-zero, the matrix multiplication in Equation (1) will be bijective and thus corresponds to permuting the set \( \{0, \ldots, p^m\} \). Hence a set of generator matrices with non-zero determinants determines a point set by Equation (2) that is perfectly stratified in each single dimension \( i \) and thus a Latin hypercube sample. The most prominent example of this property is the Sobol’ sequence.

As a generator matrix with non-zero determinant implies perfect stratification, we conclude that non-zero determinants may be used to express uniformity constraints.

4.2 Design Constraint: Stratification and Net Properties

The intuition developed in Section 4.1 may be used to define uniformity constraints across dimensions. Let \( K = (k_0, \ldots, k_{s-1}) \in \mathbb{N}^s \) be a set of non-negative integers such that \( \sum_{i=0}^{s-1} k_i = m \). Then, for each \( k_i \) the first rows of \( C_1 \) map the index \( a \) to one of the intervals \( [1 \cdot p^{m-k_i}, (l+1) \cdot p^{m-k_i}) \), \( 0 \leq l < p^{k_i} \). Composing all \( k_i \) first rows of the corresponding generator matrix \( C_i \) results in an \( m \times m \) matrix \( M_K \) that maps the index \( a \) to one of \( p^m \) elementary intervals of size \( p^{m-k_i} \times \cdots \times p^{m-k_{s-1}} \) as illustrated for two dimensions in Figure 1.

By analogy with Section 4.1, a composite matrix \( M_K \) with a non-zero determinant is a bijection between the index \( a \) and the elementary intervals specified by \( K \). In case the dimension \( s \) divides the number \( m \) of digits, the generator matrices \( C_0, \ldots, C_{s-1} \) generate a stratified point set if the determinant

\[
\det(M_{(m/s,\ldots,m/s)}) \neq 0,
\]

because each stratum of volume \( p^{m/s} \times \cdots \times p^{m/s} \) contains exactly one point. In case \( s \) does not divide \( m \), the property can be generalized to elementary intervals with a ratio of at most \( p \) between the longest and shortest dimensions. This leads to

**DESIGN CONSTRAINT 1 (GENERALIZED STRATIFICATION).** A point set of size \( p^m \) is said to be stratified if

\[
\forall K = (k_0, \ldots, k_{s-1}) \in \left\{ \left\lfloor \frac{m}{s} \right\rfloor \times \left\lfloor \frac{m}{s} \right\rfloor \right\}^s \text{ with } \sum_{i=0}^{s-1} k_i = m,
\]

all cells of size \( p^{m-k_0} \times \cdots \times p^{m-k_{s-1}} \) contain exactly one point. Thus \( C_0, \ldots, C_{s-1} \) generate a stratified point set if and only if

\[
\det(M_K) \neq 0. \tag{3}
\]

Classic stratification along the canonical axes is a special case of the more general concept of stratification by elementary intervals. The constraint resulting from the definition of \( (0,m,s) \)-nets \cite{Niederreiter1992} is:

**DESIGN CONSTRAINT 2 ((0,M,S)-NET).** An \( s \)-dimensional point set of size \( p^m \) is said to be a \((0,m,s)\)-net if

\[
\forall K = (k_0, \ldots, k_{s-1}) \in \mathbb{N}^s \text{ with } \sum_{i=0}^{s-1} k_i = m,
\]

all cells of size \( p^{m-k_0} \times \cdots \times p^{m-k_{s-1}} \) contain exactly one point. Thus \( C_0, \ldots, C_{s-1} \) generate a \((0,m,s)\)-net if and only if

\[
\det(M_K) \neq 0. \tag{4}
\]

We conclude that stratification and \((0,m,s)\)-net constraints of digital nets and sequences can be verified by computing determinants of matrices that are composed from sets of first rows of generator matrices \( \{C_0, \ldots, C_{s-1}\} \) in \( \mathbb{F}_p \).

4.3 Design Constraint: Sequences

Many applications use adaptive sampling, where the number of samples is controlled by a termination criterion, e.g. a threshold on the variance in Monte Carlo integration. Progressive random sampling is straightforward, as it just requires to draw more random samples. In the case of our digital construction, points have to respect uniformity constraints that are only satisfied for sample counts of powers of \( p \), see Section 3.2.

In theory a sequence property shall be valid for an infinite number of samples. In practice the number of samples is bounded due to finite floating point precision and finite computation time. Given the base \( p \), this allows us to select a suitable \( m \) determining an upper bound of \( p^m \) samples that covers the range of samples representable in floating point arithmetic. As a consequence, one can select a point set of maximum size \( p^m \) and consider it to be sequential if, for \( 1 \leq i \leq m \), all subsets of the first \( p^i \) points satisfy the

**DESIGN Constraint 3 (SEQUENTIAL PROPERTY).** \( C_0, \ldots, C_{s-1} \) have a sequence property if the property is valid for all \( j \times j \) submatrices \( C_{j0}, \ldots, C_{j_{s-1}} \) of \( C_0, \ldots, C_{s-1} \) anchored at the most significant digit, \( 1 \leq j \leq m \).

4.4 Design Constraint: The Prime Base \( p \)

The base \( p \) in which one decomposes the index \( a \) in Equation (1) is an important choice to consider in the design of generator matrices. It is a known result \cite{Niederreiter1992} that in base \( p \) it is neither possible to construct a \((0,m,s)\)-net for \( s > p + 1 \) nor a \((0,s)\)-sequence for \( s > p \). The base may also have an impact on the satisfiability of the entire set of other constraints. It determines the number of values any specific matrix entry can take. In turn, each constraint \( \det(\cdot) \neq 0 \) may reduce the range of possible matrix entries.

As an example, given a matrix \( C_0 \), we search for a matrix \( C_1 \) such that both generate a \((0,2)\)-sequence in base \( p = 2 \). By induction one can show that there is only one possible value for each entry of \( C_1 \) and thus there exists exactly one \( C_1 \) that results in a \((0,2)\)-sequence given \( C_0 \). In particular, if \( C_0 \) is the identity matrix, \( C_1 \) corresponds to the Pascal triangle, and \((C_0, C_1)\) forms the first two dimensions of the Sobol’ sequence. This restriction due to the small base can be severely limiting. For instance, we show in Figure 2 the samples obtained by generalizing the construction of Cascaded Sobol’ Sampling \cite{Paulin2021} where all consecutive pairs of dimensions form \((0,m,2)\)-nets, to instead impose the constraint that all consecutive pairs
of dimensions form \((0, 2)\)-sequences, using base \(p = 2\). Although consecutive pairs of this construction are \((0, 2)\)-sequences, half of the \(2\)-dimensional projections have extremely poor uniformity that can be detrimental in many applications such as rendering. This is due to the fact that since satisfying the \((0, 2)\)-sequence constraint has only one solution in base \(p = 2\), all matrices with even indices equal \(C_0\) and others equal \(C_1\).

Selecting a higher base comes at the price of a larger distance between sample counts at which design constraints derived in the previous sections can be verified. We further discuss changing the base in Section 6.2.

5 SOLVING THE CONSTRAINTS

Given a problem dimension \(s\), a base \(p\), and set of uniformity constraints as discussed in the previous section, the goal is to determine \(s\) generator matrices \(C_i\) with entries in \(\{0, \ldots, p - 1\}\) that satisfy the constraints. In this section, we prove that these constraints are equivalent to an integer linear program [Wolsey 2020], whose solution – if existent – yields a set of generator matrices with the desired uniformity properties.

Our approach is motivated by Definition 3.3 stating that \((t, s)\)-sequences are sequences of \((t, m, s)\)-nets for \(m > t\). We hence implement matrix construction as a greedy algorithm that iterates over the number of digits \(m\) starting with \(1 \times 1\) matrices until the target size is reached. To grow the matrices, we only need to consider triangular matrices, see the proof in Appendix A:

**Theorem 5.1.** If a set of constraints can be satisfied by square matrices, it will equivalently be satisfiable by triangular matrices up to an Owen scrambling step.

All the aforementioned uniformity constraints have the common form of
\[
det(M) \neq 0. \tag{5}
\]
Using the recursive determinant formula yields a polynomial of degree \(m\) with the \(m^2\) matrix entries as variables in \(\mathbb{F}_p\) for each constraint. Already solving for polynomials on integers is an NP-hard problem with few efficient approximations [Belotti et al. 2013]. However, we can transform the Inequality (5) into a set of linear constraints to be solved in \(\mathbb{Z}\) instead of \(\mathbb{F}_p\): For \(p\) a prime number, the first linear constraint amounts to
\[
0 < \det(M) - kp < p \tag{6}
\]
with \(k \in \mathbb{Z}\) an additional variable representing the modulo arithmetic of \(\mathbb{F}_p\). As we no longer operate in \(\mathbb{F}_p\), the matrix elements need to be constrained to \(\{0, \ldots, p - 1\} \subset \mathbb{Z}\) that is
\[
\forall 0 \leq j \leq m - 1, 0 \leq M_{j, m-1} < p. \tag{7}
\]
Note that since we want to iteratively grow the matrices, the \(m - 1 \times m - 1\) triangular matrix is already known. Hence only the newly added last column of matrix entries \(M_{j, m-1}\) adds to the constraints.

The key observation is that these \(m+1\) constraints form an integer linear program that can be used to determine the next larger set of generator matrices - much in the reverse spirit of the recursive determinant formula. The resulting sequence of linear integer programming problems remains NP-hard but is much easier to solve. We formalize this in

**Theorem 5.2.** For given \(m \times m\) generator matrices \(C_0, \ldots, C_{s-1}\), adding a column of variables \(c_j^{\prime}\) to the right and adding a row of \(m\) zeros at the bottom yields a set of \((m + 1) \times (m + 1)\) matrices \(C'_0, \ldots, C'_{s-1}\), i.e.
\[
C'_i = \begin{pmatrix} C_i & c_{i, m+1} \\ 0 & \cdots & 0 & c_{i, m+1} \end{pmatrix},
\]
satisfying a set of properties that is an integer linear program.

**Proof.** Let \(L\) be the set of constraints we wish to satisfy. Each constraint is in the form of \(\det(M) \neq 0\) for a given matrix \(M\). According to our design principles \(M\) is an \((m + 1) \times (m + 1)\) matrix composed of lines from multiple \(C'_j\). Thus all cells are known except for the last column. Since the determinant is linear by columns, it is linear for all variables \(c_j^{\prime}\) in the new column, leading to the fact that Equation (6) and (7) correspond to a pair of integer linear constraints. Packing all determinants together, \(L\) defines an integer linear program.

When the constraints are transformed into an integer linear program they can be prioritized by assigning a weight \(w_j\). This is implemented by extending the hard constraint of Equation (6) to
\[
x_j \leq \det(M) - kp < p, \text{ where } x_i \in \{0, 1\}, \tag{8}
\]
and adding \(w_j x_j\) to the objective function to be maximized. Unless \(\det(M) = 0\), in which case the constraint cannot be satisfied and hence \(x_j = 0\) must follow, \(x_j\) will be set to 1 and \(w_j x_j\) will count towards the objective function.
ranges = -
Det = 1
Det = 1
Det = 1
Det = 1

ACM Trans. Graph., Vol. 41, No. 4, Article 84. Publication date: July 2022.

... ... ... ... ...

Fig. 3. Modifiers u0, u1, u2, u4 illustrate for the relaxed (0, m, z)-net constraint in 2 dimensions, base 2, and for 24 samples: u0 corresponds to a stratification test enforcing one sample per \( \left( \frac{1}{2} \times \frac{1}{2} \right) \) strata, u4 amounts to a complete (0, m, 2)-net check, and u2 leads to intermediate constraints. Such properties can be assessed by checking the determinant of the \( M_k \) matrices: 2 rows from \( C_0 \) and \( C_1 \) (u0), 1 row from \( C_0 \), three from \( C_1 \) and conversely (u2), or the full \( C_0 \) and \( C_1 \) matrices (u4). The strata depicted in the unit squares on the right are the elementary intervals underlying the displayed (0, 4, 2)-net.

In summary, we implement matrix construction as a greedy algorithm that iterates over \( m \) and digit by digit determines the columns of all \( C_l \) by solving the set of linear constraints derived in Equations (6) and (7) while maximizing a target objective function that is the weighted sum of satisfied weak constraints (see Figure 4). As solving this problem remains NP-hard, we rely on efficient heuristics as implemented in the CPLEX [IBM 2022] library in order to find instances that satisfy the constraints.

With our greedy approach it may happen, especially for restrictive sets of constraints, that choices made for a previous column may render finding the next column unsatisfiable. To tackle this issue we add a backtracking feature that when faced with an unsatisfiable set of constraints drops the most recently added column and tries again with different choices. When this strategy fails again, the algorithm starts from scratch with empty matrices. This behavior can be observed in the timings of the mixed profile in Figure 5.

If even after backtracking the linear integer programs remain infeasible, our algorithm will signal that the chosen set of constraints does not admit a solution. In this case, the user needs to consider a higher prime base or turning some of the hard constraints into weak ones. We also provide a timeout parameter that limits the maximum time spent on maximizing the number of satisfied weak constraints.

Uniformity beyond Generator Matrices. Up to this point both constraints and solver are deterministic. In order to optimize with respect to uniformity metrics that cannot be expressed by generator matrices alone, we extend the objective function by a stochastic regularization term:

\[
\max \sum_{j=1}^{V} \sum_{i=0}^{m+1} w_j x_j - \min_{j=1...V} \left( \frac{\sum_{i=0}^{m+1} c_j m - r_j}{s \cdot m \cdot (p-1)} \right)
\]

Besides maximizing the objective function by satisfying a maximum number of the \( V \) weak constraints, we generate random numbers \( r_j \in \{0, \ldots, p-1\} \) and subtract the normalized \( d_j \)-distance between them and the matrix elements of the new columns. The normalization accounts for the maximum distance, which is the product of the largest possible number \( p-1 \) and the number \( s \cdot m \) of variables to be determined by Theorem 5.2. In order not to override the weak constraints, the normalized difference is scaled by the smallest weight of the weak constraints. As a result, the solution of the ILP will be as close as possible to the random choice \( r_j \), which allows us to sample a large diversity of solutions satisfying all strong constraints, while optimizing the weak ones.

Given such a set of solutions, any application specific selection may be exercised. Options are uniformity metrics that cannot be expressed as generator matrix properties such as maximized minimum distance [Grünschloß et al. 2008], spectral analysis [Pilleboue et al. 2015], or Optimal Transport [Paulin et al. 2020].

For the example of our rendering application in Section 7.1, we generate 64 sets of matrices and select the best candidate with respect to lowest \( L_2 \) discrepancy. Usually, the quality of the generated matrices exposes low variance as can be seen Figure 6-top.

6 THE MATBUILDER SOLVER

The input to the MatBuilder program is a text file that specifies the constraints imposed on the desired point set as introduced in the
next section. The solver returns the $s$ generator matrices $C_i \in \mathbb{F}_p^{m \times m}$ that are used to generate up to $p^m$ samples in $[0,1)^s$ according to Equation (2). The performance of the algorithm is explored in Section 6.2.

6.1 Constraint Specification Language

The header of the constraint text file specifies the dimension $s$, the finite field $\mathbb{F}_p$, and the number of uniformly distributed points $p^m$ to generate:

```
$s=<\text{dimension } s \text{ of point set}>
p=<\text{base } p>
m=<\text{matrix dimension } m>
```

Hard constraints on selected dimensions are added using the keywords

```
net <i_1> <i_2> ... <i_s'> #$(\emptyset, m, s')$-net constraint
stratified <i_1> <i_2> ... <i_s'>
```

where the dimension $s'$ amounts to the number of dimensions enumerated after the keyword and

```
weak 1 net 3 5
weak 1 net 2 4
weak 1 net 1 5
weak 1 net 1 4
weak 1 net 1 3
weak 1 net 1 2
weak 1 net 0 4
weak 1 net 0 3
weak 1 net 0 2
weak 1 net 0 1
```

By default, the `net` and `stratified` constraints are enforced for submatrices, i.e. all $m' < m$, which allows for progressive sampling with the selected constraints guaranteed. As for example, it does not make sense to check the stratification for less than $p^2$ points in dimension $s = 2$, using the `from` and `to` keywords, constraints can be restricted to sample indices between $p^{m'}$ and $p^m$:

```
from <m_1> to <m_2> net <i_1> <i_2> ... <i_s'>
from <m'> stratified <i_1> <i_2> ... <i_s'>
```

We have added a modifier to the `net` keyword, denoted `u<m'>` for $0 \leq m' < m$ to define relaxed versions of the `net` property: we only keep constraints that rely on an $M_{m'}$ matrix (see design constraint 2) with a maximum difference of $m'$ between the number of rows taken from the $C_i$. This way $u\emptyset$ corresponds to the Property A of [Sobol’ 1967] (contiguous blocks of $p^s$ samples are stratified), $u1$ corresponds to design constraint 1, $u\emptyset$ would correspond to the original `net` constraint (see Figure 3). This modifier allows for a control of both the positively correlated uniformity strength and computational cost of the problem. The efficiency of different values $m'$ with respect to discrepancy is shown in Figure 6.

When hard constraints cannot be satisfied, the `weak` keyword allows one to attach a weight resulting in a weak constraint:

```
weak <w> from <m_1> to <m_2> net <i_1> <i_2> ... <i_s'>
weak <w> from <m'> stratified <i_1> <i_2> ... <i_s'>
```

6.2 Numerical Evidence

To evaluate the performances of the MatBuilder program, we consider several generic scenarios: first, a low discrepancy sequence in base 3 with up to $3^{10}$ samples in dimension 8, with as low discrepancy as possible for the full dimension (net):

```
Generic-full-space-LDS
s=8
p=3
```

Solutions to this profile will be as close as possible to $(0,8)$-sequences up to $3^{10}$ samples (note that exact $(0,s)$-sequences do not exist for $s > p$ in base $p$ [Niederreiter 1992]).

Second, we consider a perfect projective profile in dimension 6, where each consecutive pair of dimensions is a $(0,2)$–sequences up to $3^{10}$ in base $3$, and all remaining pairs must be as close as possible to a generalized stratification (see Figure 7 for a comparison to the Sobol’ sequence, and the net obtained by LatNetBuilder [L’Ecuyer and Munger 2016], when optimizing the same pairs of projection):

```
Generic-proj-LDS
s=6
p=3
m=10
```

Third, we take a look at an Orthogonal Array profile in dimension 9: we search for a sequential sampler in base 3 with OA strength 3 (each triplet of dimension are stratified starting from $3^3$ samples), and a final weak stratification constraint in 9 dimensions:

```
Generic-OA
s=9
p=3
m=10
```

Finally, a more complex scenario combines stratified and net properties in dimension 10 for $p = 3$:

```
Mixed
s=10
p=3
m=10
```

```
net 0 1
net 1 2
net 2 3
net 3 4
net 4 5
weak 1 net 0 2
weak 1 net 0 3
weak 1 net 0 4
weak 1 net 0 5
weak 1 net 1 3
weak 1 net 1 4
weak 1 net 1 5
```

```
stratified 0 1 2 3 4 5 6 7 8
```

```
```
Figure 5: Performance: On the top we show the timings of the MatBuilder solver as a function of the matrix size \( m \). For the mixed profile, the plots show several backtracks for small \( m \) before being able to expand the matrices \((u_4\text{ and } u_7\text{ have the same timings}). At the bottom, the size of the reduced integer linear programs are shown during execution (number of rows and number of nonzero entries) for the generic-full-space-LDS profile (most difficult one in our tests). For this profile, up to 60 calls to the internal integer linear program simplification procedure have been performed.

7 APPLICATIONS

We demonstrate the advantages of samplers generated from application specific constraints to master uniformity over projections for path tracing, texture exploration, and optimal control.

7.1 Path Tracing

In rendering, the path space exposes a clear nested structure: each high-dimensional light transport path is built from low-dimensional events: sampling a pixel footprint (2D), sampling the lens aperture (2D), sampling the time (1D), sampling a point on a light source (3D, usually 1D to select a light source and another 2D to sample a point on this light source), sampling a direction scattered from a material (3D, usually 1D to select a layer and another 2D to sample a direction from this layer), and so on. Previous work [Kollig and Keller 2002; Schlick 1991] has composited high-dimensional path space samples from low-dimensional samplers connected by some form of randomization.

We design a path-tracing profile PT-Profile to leverage perfect low dimensional stratification and good stratification over increasing dimensions to replicate the path space structure:

<table>
<thead>
<tr>
<th>PT-Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>p=3</td>
</tr>
<tr>
<td>s=6</td>
</tr>
<tr>
<td>m=12</td>
</tr>
<tr>
<td>net 0 1</td>
</tr>
<tr>
<td>net 1 2</td>
</tr>
<tr>
<td>net 2 3</td>
</tr>
<tr>
<td>net 3 4</td>
</tr>
<tr>
<td>net 4 5</td>
</tr>
<tr>
<td>weak 1 net u4 0 1 2</td>
</tr>
</tbody>
</table>
7.2 Exploring Parametric Materials

In this scenario, the objective is to sample the space of a parametric material with explicit correlation between some parameters. This is an initial step in (semi-)procedural texture exploration or texture space analysis [Guehl et al. 2020; Lasram et al. 2012a,b]. As an example, we consider a procedural texture with seven dimensions to define brick pattern, window brace, and lintel. The structure of the parameters leads to the following profile:

\[
\begin{align*}
&s=7 \\
&p=2 \\
&m=5 \\
# (\text{Brick-
Amount-X, Brick-
Amount-Y}) \\
# \text{net} 0 1 \\
# (\text{Window-
Brace-
Amount-X, Window-
Brace-
Width-X,} \\
# \text{Window-
Brace-
Amount-Y, Window-
Brace-
Width-Y}) \\
# \text{weak } 1 \text{ net } 3 4 5 6 \\
# \text{weak } 2 \text{ net } 0 1 2 3 4 5 6 \\
# \text{Overall uniformity, including Brick-Lintel-Width} \\
\end{align*}
\]

The constraints ensure the \((0, m, 2)\)-net stratification of the first pair of dimensions, a weak 4-dimensional net property of the last four dimensions, and an overall net property across all seven dimensions. Figure 9 illustrates the first 16 realizations as compared to a uniform random sampling of each parameter, exhibiting a more uniform exploration of the parameter space. For this profile, matrices are generated in 3.7 seconds by our MatBuilder software.

7.3 Optimal Control

Reinforcement learning can be used to control a robot, a virtual character, or a dynamic system. This amounts to discretizing the states of the system as well as possible actions applied to it, defining a reward for achieving particular states, and iteratively solving Bellman equation through value iterations. This results in a value function and optimal control policy at each state, which can then be interpolated during the simulation in real-time to control the system. We apply the technique described by Coros et al. [2009] to control a simple inverted pendulum on a chart to reach an upward orientation, a specific position, and a near zero velocity and angular velocity. The state space is 4-dimensional (position \(x\), velocity \(\dot{x}\), angle \(\theta\),
and angular velocity \( \dot{\theta} \), and the action space is 1-dimensional (a horizontal force \( u \)). The action space is uniformly discretized. We compare state-of-the-art samplings of the state space with samples created by MatBuilder with the following profile that seeks 4-d uniformity:

![Matrix diagram](image)

**Fig. 7. Projective LDS profile:** (a) Projection of the first 2048 Sobol’ samples in dimension 6; (b) projections obtained from optimized LatNetBuilder matrices using a projective profile (see Appendix B); (c) the first \( 3^7 = 2187 \) samples for the matrices obtained with the generic-\textsc{proj}-LDS profile exhibiting better uniformity in the 2D projections. In (d), we quantify our benefit through a 2D discrepancy comparison for all projections. The lower is discrepancy, the better is uniformity.

The state-of-the-art samplers we experimented with are random sampling, a Sobol’ sequence [Joe and Kuo 2008; Sobol’ 1967], and a rank-1 lattice produced by LatNetBuilder [L’Ecuyer and Munger 2016]. In all cases, we bias these samples so that they follow the same Gaussian distribution around the target state and get a finer discretization near the solution. In practice, we used the inverse cumulative density function of the Gaussian distribution to generate \( 3^{12} = 531,441 \) samples (or \( 2^{19} = 524,288 \) for Sobol’ in base 2), plus an additional sample placed at the exact desired state. To compare these discretizations, we simulate 200 trajectories for 2000 time steps each and compute the average number of time steps \( \alpha \) the pendulum is within an area of non-zero reward (i.e., within the area considered as success), the number of trajectories \( \beta \) which succeed at least in one time step to get a non-zero reward, and the average value \( \gamma \) of the value function (higher values mean that more time is spent near the desired state). For all these metrics, higher values indicate better behavior. For our stratified sampling, \( \alpha = 170, \beta = 200 \) and \( \gamma = 530 \) (the reward for achieving success is 1000, and the maximum value of the optimized value function is near 2500). For 200 trajectories with a single realization of a random sampling, \( \alpha = 12, \beta = 141 \) and \( \gamma = 349 \). For this setup, the Sobol’ sequence yields \( \alpha = 7, \beta = 44 \), and \( \gamma = 611 \) while the rank-1 lattice yields \( \alpha = 1, \beta = 20 \), \( \gamma = 189 \). For 200 trajectories consisting in one (different) trajectory for 200 random sampling realizations, \( \alpha = 30, \beta = 121 \) and \( \gamma = 594 \). Figure 10 shows a \((x, \theta)\) slice of 3369 samples of our \( 3^{12} \) samples within a small \((x, \theta)\) interval, as well as 3 representative trajectories. Clearly, the sampler generated by MatBuilder excels.

8 DISCUSSION AND PERSPECTIVES

Our algorithm is greedy with respect to the size of matrices and uniformity properties consider whole matrix rows at a time (and not just the last column). Consequently, some solution may be exponentially hard to find as they may require a specific succession of choices among a wide set of locally satisfactory ones. It is interesting to search for a solution generating lines of a matrix at a time. Furthermore, rather than using a greedy algorithm iterating integer linear programs, one may opt for an algorithm generating matrices one at a time through a non-linear integer program solver, which may work better in this aspect.

The design of the objective function of our integer linear programs implies a relatively small control over the way how weak sub-properties of net constraints are distributed. The solver may find the most optimal solution to be satisfying many of the constraints for a single group of dimensions while leaving a pair of dimensions entirely correlated. In perspective, this may be solved using a non-linear convex objective that is only be slightly slower to solve.

In our paper we only deal with prime Galois fields, since it relies on modular arithmetic that can easily be translated into linear constraints in \( \mathbb{F}_p \). This technique cannot be performed for powers of prime bases as they define a specific set of arithmetic operators. To extend prime Galois fields to power bases would require a solver that is able to process addition and multiplication tables specific to the individual Galois fields over prime power bases.

In our rendering section we discuss the fact that the base of the generator matrices sometimes have a huge impact on the quality of integration. The reason for this is that our selectors split the point set at separations of strata considered by the net property and we
Fig. 8. **Rendering results:** In 6 dimensions, the dimensions are used to sample the lens (2D), the time (1D), and the light sources (3D). For light sources, the first dimension is used to select a light source (2, 3, or 5 light sources for rows 1, 2, and 3), the remaining 2D coordinates sample a point in the selected light quad. In 8 dimensions, we sample the pixel (2D), the direct lighting (3D) and the indirect lighting (3D). For sampling indirect bounce directions, the first dimension is used to select a material component, and the remaining 2D coordinates importance sample a direction. The first column shows the ground-truth images, the second one a per pixel labelling according to the sampler (Random, ZeroTwo, Sobol’, Cascaded, and our PT-Profile) with lowest error at 243spp for our sampler in base 3 or 256spp for others. Finally the last column presents the mean squared error (MSE) for various sample counts.
9 CONCLUSION

Highly uniform point sets can be tailored specific to the structure of an application as a set of constraints that can be transformed into an integer linear program. Solving such an integer linear program yields a construction for samples that in specific applications have been demonstrated to outperform classic constructions. We hence introduced a new and powerful way for the computer-aided design of generator matrices for digital nets and sequences.

ACKNOWLEDGMENTS

This work was partially funded by ANR-16-CE33-0026 (CALiTrOp) and ANR-20-CE45-0025 (MoCaMed).

REFERENCES


A PROOF OF THEOREM 5.1

Proof. Owen scrambling [Owen 1998] is a randomization technique that does not alter the structure of the elemental intervals and hence preserves all properties of a digital net. To do so, it applies a bijective function \( \phi_{b_0, b_1} : \mathbb{P} \to \mathbb{P} \) to digit \( b_1 \) of the vector \( b \) in Equation (1). Given factors \( w_{ij} \in \mathbb{P} \), bijective functions of the form \( \phi_{b_0, b_1} (b) = \sum_{j=0}^{i-1} w_{ij} b_j + b_1 \) are a realization of Owen scrambling.

Let \( C \) be the generator matrix of size \( m \) that generated \( b \) and denote its rows by \( C^t \). Row \( C^t \) encodes the digit \( b_1 \). Hence one realization of Owen scrambling is the application of the function \( f_{2m, c^t}(C^t) = \sum_{j=0}^{i-1} w_{ij} C^t + C^t \) that acts on rows the same way as \( \phi_{b_0, b_1} \) acts on digits.

We apply \( m \) such linear functions to each row of \( C \). By definition, Gaussian elimination can be written as such a set of linear transformations, defining the factors \( w_{ij} \). Thus for any \( C \) we can find a realization of Owen scrambling that transforms it into an upper triangular matrix \( C^t \). Since all these functions \( f \) are bijective, their inverse defines one realization of an Owen scrambling that transforms \( C^t \) into \( C \).

It follows that all nets generated by any generator matrices can also be generated by upper triangular matrices with an additional scrambling step in the sense of Owen. It is therefore sufficient to only consider triangular matrices.

B LATNETBUILDER PROJECTIVE PROFILE

For Figure 7, we have used the following configuration for LATNETBUILDER to generate optimized matrices:

```
latnetbuilder --p projext --set-type net \
   --construction explicit --s 2*16 \n   --dimension 6 --norm-type 1 \
   --explanation-method random --CBC:100000 \n   --figure-of-merit projdep:t-value \n   --weights order-dependent:0:1,1 \n   --multilevel true --combiner sum
```