Generator Matrices by Solving Integer Linear Programs

Lois Paulin, David Coeurjolly, Nicolas Bonneel, Jean-Claude Iehl, Victor Ostromoukhov, Alexander Keller

Generator Matrices by Solving Integer Linear Programs

Loïs Paulin, David Coeurjolly, Nicolas Bonneel, Jean-Claude Iehl, Victor Ostromoukhov, and Alexander Keller

Abstract In quasi-Monte Carlo methods, generating high-dimensional low discrepancy sequences by generator matrices is a popular and efficient approach. Historically, constructing or finding such generator matrices has been a hard problem. In particular, it is challenging to take advantage of the intrinsic structure of a given numerical problem to design samplers of low discrepancy in certain subsets of dimensions. To address this issue, we devise a greedy algorithm allowing us to translate desired net properties into linear constraints on the generator matrix entries. Solving the resulting integer linear program yields generator matrices that satisfy the desired net properties. We demonstrate that our method finds generator matrices in challenging settings, offering low discrepancy sequences beyond the limitations of classic constructions.

Key words: Quasi-Monte Carlo methods · Low discrepancy sequences · Digital (t,m,s)-nets and (t,s)-sequences · Generator matrices · Optimization · Integer linear programs

1 Introduction

Monte Carlo and quasi-Monte Carlo integration are an important part of many numerical algorithms, for example, in computational finance and image synthesis. Problems in these fields often have an intrinsic structure that allows one to benefit

Loïs Paulin e-mail: lois.paulin@ens-lyon.fr · David Coeurjolly e-mail: david.coeurjolly@cnrs.fr · Nicolas Bonneel e-mail: nicolas.bonneel@liris.cnrs.fr · Jean-Claude Iehl e-mail: jean-claude.iehl@liris.cnrs.fr · Victor Ostromoukhov e-mail: victor.ostromoukhov@liris.cnrs.fr
Université de Lyon, UCBL, CNRS, INSA Lyon, LIRIS

Alexander Keller e-mail: akeller@nvidia.com
NVIDIA, Fasanenstr. 81, 10623 Berlin, Germany

from a particular uniformity profile imposed on selected subsets of dimensions. This uniformity may be specified by (t, m, s)-net properties, i.e. low discrepancy properties. A popular way to compute low discrepancy sequences is via generator matrices. Finding generator matrices that match a specific uniformity profile has been a challenging problem. Low discrepancy sequences, such as Sobol', Halton, and others, use specific matrix constructions to achieve provably strong properties. However, these constructions represent only a tiny fraction of all possible generator matrices and hence may not be able to ideally match a desired uniformity profile. As an example, Joe and Kuo's work [3] on optimizing the uniformity of all pairs of dimensions demonstrates that the resulting generator matrices do not reach a satisfactory 2D uniformity. Methods for optimizing general matrices have been devised [1]. They often rely on the random generation and selection of generator matrices. This approach fails when the desired uniformity profiles are too restrictive. Furthermore, some sets of uniformity profiles are provably infeasible in small prime bases such as $b = 2$.

We propose a linear algebra-based understanding of the fundamental theorems of digital nets and their generator matrices. We use this understanding to convert projective uniformity properties into polynomial and linear constraints on generator matrix entries. Numerically solving for these constraints allows us to construct generator matrices with net structures that have not been achieved before. Our method works in higher prime bases, too, offering one more degree of freedom to satisfy a specific set of projective net properties.

2 Quasi-Monte Carlo Methods

Quasi-Monte Carlo [7] integration estimates the value of the integral of f by

$$\int_{[0,1)^s} f(\mathbf{u}) d\mathbf{u} \approx \frac{1}{N} \sum_{i=0}^{N-1} f(\mathbf{x}_i) \,.$$

Rather than using independent realizations of uniformly, identically distributed random variables, low discrepancy sequences are used to generate deterministic samples $\mathbf{x}_i$ that are much more uniformly distributed over the s-dimensional unit cube $[0, 1)^s$ than uniform random samples can be.

The uniformity of a point set $P = \{\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}\}$ can be measured by the star-discrepancy [7, p. 14]

$$D_N^*(P) := \sup_B \left| \lambda_s(B) - \frac{1}{N} \sum_{i=0}^{N-1} c_B(\mathbf{x}_i) \right| \,,$$

where the supremum is taken over all sets B of the form $B = \prod_{i=1}^{s} [0, b_i]$, with $0 \leq b_i \leq 1$, λ_s is the Lebesgue measure of B, and c_B is the characteristic function of the set B that is one for $\mathbf{x}_i \in B$ and zero otherwise. The star-discrepancy can be

interpreted as the worst-case integration error of the class of characteristic functions c_B. The more uniformly the points are distributed, the smaller the star-discrepancy is. Informally, a point set is called low discrepancy if its star-discrepancy is small. For low-discrepancy sequences of points, the star-discrepancy vanishes in $O\left(\frac{\log^s N}{N}\right)$, while random points can only achieve a discrepancy of order $O\left(\sqrt{\frac{\log\log N}{N}}\right)$.

For functions of bounded variation V in the sense of Hardy and Krause, the Koksma-Hlawka inequality

$$\left|\frac{1}{N}\sum_{i=0}^{N-1} f(\mathbf{x}_i) - \int_{[0,1)^s} f(\mathbf{u})d\mathbf{u}\right| \leq V(f)\cdot D_N^*(P)\,,$$

bounds the integration error by the product of variation and star-discrepancy. Even though in practice many functions may not be of bounded variation, the inequality suggests that increasing the uniformity of the samples by decreasing the discrepancy may lower the integration error.

2.1 Digital (t,s)-Sequences and (t,m,s)-Nets

Deterministic digital (t,s)-sequences and (t,m,s)-nets [7] are low discrepancy point sequences and sets that can be efficiently generated. Given generator matrices $C_1,\ldots,C_s$, the algorithm

$$x_i^{(j)} = \begin{pmatrix} b^{-1} \\ \vdots \\ b^{-m} \end{pmatrix}^T \cdot \underbrace{C_j \cdot \begin{pmatrix} i_1(i) \\ \vdots \\ i_m(i) \end{pmatrix}}_{\text{multiplication in } \mathbb{F}_b} \tag{1}$$

first multiplies each generator matrix C_j by the point index $i = \sum_{k=1}^{\infty} i_k(i)b^{k-1}$ represented as vector of m digits i_k in the integer base b. The result of the matrix multiplication in a field $\mathbb{F}_b$ is then mapped to the unit interval $[0,1)$ by a scalar product with the vector of inverse powers of the base b. The stratification and hence uniformity properties of this digital construction are characterized by elementary intervals as given by

Definition 1 (see [7, p. 48]) An interval of the form

$$E := \prod_{j=1}^{s} \left[a_j b^{-d_j}, (a_j+1)b^{-d_j}\right) \subseteq [0,1)^s\,,$$

for $0 \leq a_j < b^{d_j}$ and integers $d_j \geq 0$ is called an *elementary interval in base b*.

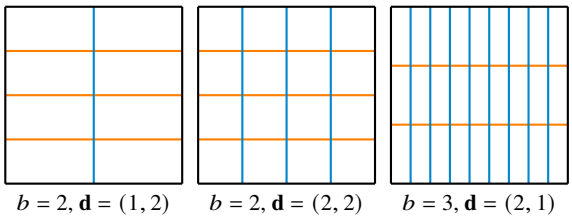


Fig. 1 Examples of elementary intervals in bases $b = 2$ and $b = 3$ in $s = 2$ dimensions. The vectors $\mathbf{d} = (d_1, d_2)$ determine the resolutions b^{-d_j} along the canonical axes, resulting in the depicted stratification.

The elementary intervals (see Fig. 1) are used to characterize the stratification of point sets as specified in

Definition 2 (see [7, 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 elementary interval E with volume b^{t-m}.

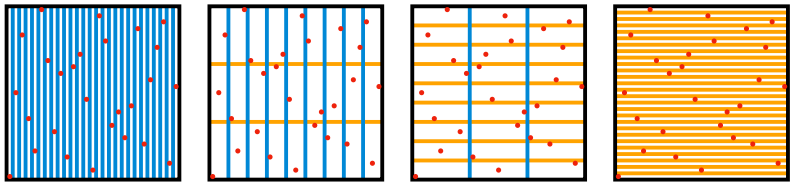


Fig. 2 Example of a $(0, 3, 2)$-net in base $b = 3$. There are $b^m = 3^3 = 27$ samples in $s = 2$ dimensions and all the elementary intervals (subdividing the x and y coordinates into $b^i = 3^i$, $i = 1..m$, intervals) contains exactly $b^t = 3^0 = 1$ point.

Fig. 2 illustrates the structure of the elementary intervals for the example of a $(0, 3, 2)$-net in base $b = 3$. Note that $\sum_{j=1}^{s} d_j = m - t$ by the definitions of the elementary intervals and the (t, m, s)-nets. The structure of the (t, m, s)-nets lends itself to the generalization to low discrepancy sequences:

Definition 3 (see [7, Def. 4.2]) For an integer $t \geq 0$, a sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ 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 $\mathbf{x}_{kb^m}, \ldots, \mathbf{x}_{(k+1)b^m-1}$ is a (t, m, s)-net in base b.

2.2 Relating Generator Matrices and Elementary Intervals

Fig. 3 illustrates the relationship between generator matrices and elementary intervals: multiplying a generator matrix C_j by an index vector (see Eqn. (1)), the

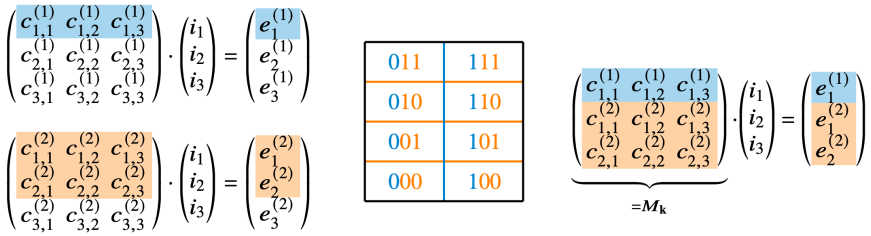


Fig. 3 Illustration of the algebraic relationship of generator matrices and elementary intervals. Left: As highlighted, the first few rows of a generator matrix determine the most significant digits e_k. Middle: For the displayed set of elementary intervals, the digit $e_1^{(1)}$ determines the partition along the first dimension, while $e_1^{(2)}$ and $e_2^{(2)}$ select the partition along the second dimension. Right: If now $\det M_{\mathbf{k}} \neq 0$, the composite matrix $M_{\mathbf{k}}$ will be a bijection between the elementary intervals and the index (i_1, i_2, i_3).

k most significant digits of the result are determined by the first k rows of the generator matrix. Then, a matrix $M_{\mathbf{k}}$ composed of the first k_j rows of C_j, where $\mathbf{k} := (k_1, \ldots, k_s) \in \mathbb{N}_0^s$ and $\sum_{j=1}^{s} k_j = m$, defines a mapping from the indices to elementary intervals of size $\left(\frac{1}{b^{k_1}}, \ldots, \frac{1}{b^{k_s}}\right)$ and consequently volume $\frac{1}{b^m}$. If and only if $\det M_{\mathbf{k}} \neq 0$ the mapping will be bijective. Hence the determinants of the matrices $M_{\mathbf{k}}$ can be used to verify the properties of $(0, m, s)$-nets, where each elementary interval contains exactly one point.

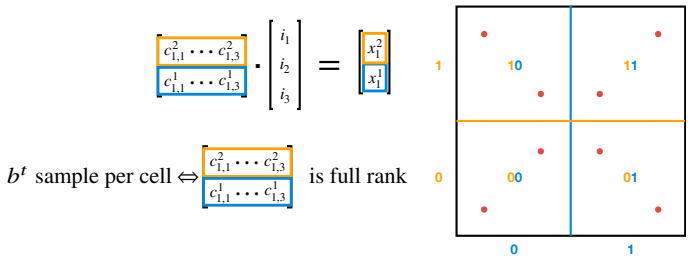


Fig. 4 Verification of the (t, m, s)-net properties for $t \geq 0$.

Considering values of $t > 0$, we select $\mathbf{k} := (k_1, \ldots, k_s)$, now with $\sum_{j=1}^{s} k_j = m - t$. This results in rectangular matrices $M_{\mathbf{k}}$ defining a mapping between indices and elementary of volume $\frac{1}{b^{m-t}}$. We want each elementary interval to be the image of exactly b^t indices. This is equivalent to having $\dim(\ker M_{\mathbf{k}}) = t$ and $\dim(\operatorname{im} M_{\mathbf{k}}) = m - t$, because the dimension of the index space is m. $M_{\mathbf{k}}$ having $m - t$ rows, this means $M_{\mathbf{k}}$ needs to have full rank. As such, the generator matrices $C_1, \ldots, C_s$ define a (t, m, s)-net if and only if $\forall k = (k_1, \ldots, k_s)$ with $\sum_{j=1}^{s} k_j = m - t$, $M_{\mathbf{k}}$ is of full rank as illustrated in Fig. 4.

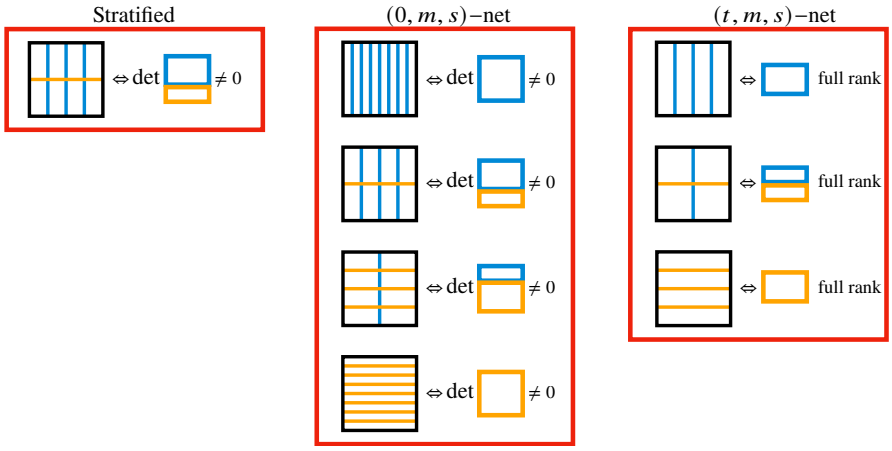


Fig. 5 Stratification properties (left) expressed by matrix determinants allow for checking $(0, m, s)$-net properties (middle) and (t, m, s)-net properties for $t > 0$. The red boxes group sets of constraints.

Fig. 5 summarizes the stratification constraints that can be specified by using matrices composited from generator matrices as described before. Regarding (t, s)-sequences, verifying the (t, m, s)-net properties for all $m \geq t$ is equivalent to checking whether all top-left square sub-matrices generate (t, m, s)-nets. This result is stated in a slightly different manner by

Theorem 1 (see [7, p. 73]) *Suppose that the integer $t \geq 0$ satisfies the following property: For any integers $m > t$ and $k_1, \ldots, k_s \geq 0$ with $\sum_{j=1}^{s} k_j = m - t$ and any $e_i^{(j)} \in \mathbb{F}_b$, the system of $m - t$ linear equations*

$$\sum_{r=0}^{m-1} c_{i,r}^{(j)} z_r = e_i^{(j)} \text{ for } 1 \leq i \leq d_j,\ 1 \leq j \leq s$$

in the unknowns $z_0, \ldots, z_{m-1}$ over $\mathbb{R}$ has exactly b^t solutions. Then the sequence $\mathbf{x}_i$ generated by Eqn. (1) is a (t, s)-sequence in base b.

(t, s)-sequences in base $b = 2$, as for example the construction by Sobol' [9] are very popular, because they can be computed very efficiently using bit-parallel vector operations for the implementation of the Galois field $\mathbb{F}_2$. The small basis, however, may be limiting, as we claim in

Theorem 2 *There are no more than b generator matrices in base b such that all pairs of matrices generate a $(0, 2)$-sequence.*

A constraint graph G is a graph with a vertex for each dimension, and with an edge between two dimensions that can be found in the same sequence constraint. As $(0, 2)$-sequences constraints between all pairs of dimension are included in (t, s)-sequence constraints, Thm. 2 states that G admitting not having any clique of size

greater than b and thus admitting a b coloring is a necessary condition for the system of constraints to have solutions. To prove the theorem, we need the following

Lemma 1 *In base b there are no more than b vectors of dimension 2 with a non-zero first component such that all vectors are pair-wise linearly independent.*

Proof Without loss of generality, we can assume that the first component of our vectors is always 1 as any vector is linearly equivalent to one with the first component equal to 1. For all $k_1 \neq k_2$ the vectors $(1, k_0)$ and $(1, k_1)$ are linearly independent. There are b possible values for the second component. Thus there are no more than b vectors of dimension 2 with a non-zero first component such that all vectors are pair-wise linearly independent. □

With the lemma at hand, we are ready to prove Thm. 2:

Proof Let $C_1, \ldots, C_s$ be s generator matrices in base b such that all pairs of matrices generate a $(0, 2)$-sequence. Then all pairs of top-left cornered square submatrices of size m generate a $(0, m, 2)$-net. For $m = 1$ this means for $1 \leq j \leq s$ all $c_{1,1}^{(j)} \neq 0$. For $m = 2$ this means that all the first rows of each matrix must be pair-wise linearly independent. By Lemma 1, there are no more than b linearly independent first rows. Thus there are no more than b matrices in base b such that all pairs of matrices generate a $(0, 2)$-sequence. □

```
#Generic-full-space-LDS
s=8
b=3
m=10
weak 1 net 0 1 2 3 4 5 6 7
```

```
#Generic-OrthogonalArrays
s=9
b=3
m=10
from 3 stratified 0 1 2
from 3 stratified 1 2 3
from 3 stratified 2 3 4
from 3 stratified 3 4 5
from 3 stratified 4 5 6
from 3 stratified 5 6 7
from 3 stratified 6 7 8
from 5 weak 1 stratified 0 1 2 3 4 5 6 7 8
```

```
#Mixed
s=10
b=3
m=10
net 0 1
net 1 2
net 2 3
net 3 4
net 4 5
from 3 stratified 0 1 2
from 4 stratified 1 2 3
from 3 stratified 2 3 4
from 4 stratified 3 4 5
from 4 to 6 stratified 0 1 2 3 4
from 4 to 6 stratified 1 2 3 4 5
```

```
#Path tracing profile
b=3
s=6
m=12
net 0 1
net 1 2
net 2 3
net 3 4
net 4 5
weak 1 net u4 0 1 2
weak 1 net u4 1 2 3
weak 1 net u4 2 3 4
weak 1 net u4 3 4 5
weak 1 net u2 0 1 2 3
weak 1 net u2 1 2 3 4
weak 1 net u2 2 3 4 5
weak 1 net u2 0 1 2 3 4
weak 1 net u2 1 2 3 4 5
weak 1 net u4 0 1 2 3 4 5
```

Fig. 6 Examples of profiles in [8], where only $t = 0$ was supported. The parameter s refers to the problem dimension, p is the prime base, and m is the matrix size to generate up to p^m points. The remaining lines specify net and stratification constraints on subsets of problem dimensions. The keyword `weak` indicates that the following constraint is not strict, and its strength w_{v_j} is given by the value next to the keyword. `from` m_1 `to` m_2 indicates a constraint only valid for the matrix sizes ranging from m_1 to m_2, hence affecting the samples in the range of indices from p^{m_1} to p^{m_2-1} ($u2$ or $u4$ only enforces subsets of constraints, more details in [8]).

3 Specifying Generator Matrices by Constraints

The profile language as introduced in [8] allows one to specify generator matrices by constraints (see Fig. 6). A profile first sets the dimension s, prime base p, and matrix size m, followed by a list of constraints. Each constraint affects a selected subset of dimensions, which allows for forging application-specific uniformity properties. Then the constraints are transformed into an Integer Linear Program (see Fig. 7) whose solution specifies the s generator matrices of size $m \times m$.

We extend the syntax of the profile language to include $(t, m, s)-$net constraints with arbitrary $t \geq 0$ as an additional parameter to the `net` and `weak net` keywords. For example, if we want to design a progressive net in dimension 6 and base 3 up to 3^{10} samples that is a $(0, 10, 2)$-net for the first two pairs of dimensions, as close to a $(1, 10, 3)$-net as possible for the dimensions $\{3, 4, 5\}$ and as close to a $(2, 10, 6)$-net as possible in 6D, the profile becomes (further examples are provided in Fig. 12):

```
b=3
s=6
m=10
net t0 0 2
net t0 1 2
weak 1 net t1 3 4 5
weak 1 net t2 0 1 2 3 4 5
```

3.1 Overlapping Constraints

Theoretical results [7] state that the best possible t-value for a (t,m,s)-net is $t = s - b - 1$. However, many applications exhibit a structure where the uniformity of a subset of dimensions greatly impacts the integration error. This has already been observed in the work of Joe and Kuo [3] and L'Ecuyer et al. [6]. As the dimension of the subsets often is much smaller than the global dimension, the t-value of the subset of dimensions theoretically can be smaller than the global t-value. In our profile language, it is straightforward to specify such potentially overlapping constraints, which offers a new way to describe generator matrices.

3.2 Specifying Constraints in $\mathbb{Z}$

Solvers are usually devised to work in $\mathbb{Z}$, however, our constraints are in $\mathbb{F}_b$. In order to take advantage of existing solvers, we need to convert our constraints to $\mathbb{Z}$. In $\mathbb{F}_b$ our constraints are of the form $\sum_i w_i x_i \neq 0$. We transform them to $0 < \sum_i w_i x_i + kb < b$ with $k \in \mathbb{Z}$ an additional variable to solve for and $0 \leq x_i < b$ (see Fig. 7). This way we emulate the modulo arithmetic of prime base Galois fields. Note that this trick is limited to prime bases b, which is the reason for naming the base `p` in our constraint language.

3.3 Weak Constraints

Each net constraint on a set of dimensions is actually a set of linear sub-constraints, one per elementary interval shape. It often happens that constraints cannot be satisfied. This can be due to a too small t-value or because of conflicting overlapping constraints. To alleviate the issue, we introduce weak constraints. All constraints were crafted such that the determinant of the corresponding matrix M is strictly in the range $\{0, \dots, b-1\}$. For the j-th weak constraint, we now make these bounds depend on a variable $\nu_j \in [0, 1]$ in the following manner:

$$\nu_j \leq \sum_i w_i x_i + k_j b \leq (b-1)\nu_j \,.$$

This way, if $\nu_j = 0$ the constraint becomes $\sum_i w_i x_i + k_j b = 0$ and if $\nu_j = 1$ the constraint results to be $0 < \sum_i w_i x_i + k_j b < b$ which is equivalent to the original hard constraint. Maximizing the sum $\sum w_{\nu_j} \nu_j$, where w_{ν_j} is the weight of constraint j, thus maximizes the number of satisfied sub-constraints while allowing some of them not to be satisfied in case they are infeasible. The complete setup of the Integer Linear Program is summarized in Fig. 7.

3.4 Polynomial Integer Program

A matrix $A \in \mathbb{F}_b^{(m-t)\times m}$ has full rank if and only if at least one of the square sub-matrices A_i obtained by dropping any t columns $i = (i_1, \dots, i_t)$ of A has full rank. This means that at least one such sub-matrix has a non-zero determinant. This can be represented by the set of polynomial constraints of degree $m - t$,

$$\exists i \in \{(i_1, \dots, i_t) \in \mathbb{N}^t \mid 1 \leq i_1 < \cdots < i_t \leq m\} \quad \det(A_i) \neq 0 \,.$$

Imposing all M_k to have full rank can thus be expressed as satisfying a set of polynomial constraints on values of matrices C_j.

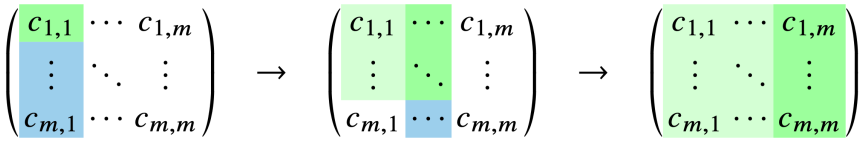
$$\forall \mathbf{k} := (k_1, \dots, k_s) \in \mathbb{N}^s, \sum_{j=1}^{s} k_j = m - t,$$
$$\exists i \in \{(i_1, \dots, i_t) \in \mathbb{N}^t \mid 1 \leq i_1 < \cdots < i_t \leq m\} \quad \det M_{k,i} \neq 0 \,.$$

with $M_{k,i}$ denoting the matrix M_k with the set of columns i removed. The set of matrices $C_1, \dots, C_s$ describes a (t,m,s)-net if and only if it satisfies a set of polynomial constraints stating that all $M_{\mathbf{k}}$ constructed from these matrices (see Sec. 2.2) have full rank.

3.5 Linear Integer Programs

Thm. 1 advocates linear constraint solving, which, however, is NP-complete. Unless $P = NP$, the computational complexity is exponential in the number of variables sm^2. In practice, such algorithms are infeasible, as the number of variables is too high. However, assuming all values in the matrices to be known up to column $i-1$ and trying to determine the i-th column simplifies the problem to that of satisfying a set of linear constraints with sm variables, since the determinant is a linear function of the matrix columns. In combination with the heuristic applied in modern Integer Linear Program Solvers [2], it becomes possible to compute tangible results.

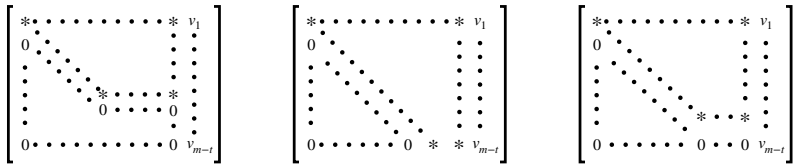
We hence use a greedy algorithm to construct the generator matrices column-by-column:



At each step, the top-left square sub-matrices are checked to guarantee the sequence property (highlighted in green). As our matrices have finite size, we ensure the Def. 3 of a (t,s)-sequence only up to b^m samples. Hence, we call point sets generated by such matrices *progressive* (t,m,s)*-nets*.

In the highlighted columns, elements in green are chosen at random according to the constraints, while the blue elements are not constrained and become selected at random.

To determine these linear constraints for each M_k, we check whether the matrices have full rank by performing a *symbolic* Gaussian elimination. Gaussian elimination seeks to triangularize a matrix by iteratively subtracting linear combinations of rows. For a rectangular matrix, the Gaussian elimination results in 3 possible outcomes:



Case (a) has multiple rows with only the last column with a non-zero value and thus cannot have full rank whichever values the different v take. Case (b) has no row with only the last column with a non-zero value and thus has full rank independently of the values the different v take. Case (c) has exactly one row with only the last column non-zero and thus has full rank iff $v_{m-t} \neq 0$. Following the Gaussian elimination

process, v_{m-t} is a linear combination of the variables of the last column of M_k with weights depending on the values of the first columns of M_k.

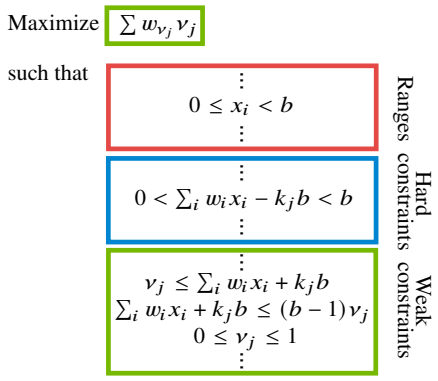


Fig. 7 Anatomy of an Integer Linear Program (ILP). Please refer to the text for details.

In summary, to grow the matrices C_i according to our greedy strategy, the values of the $c^{(i)}_{l,m+1}$ (abstracted as x_i in our formulas) in the last column of their respective C_i are determined by solving an ILP, which consists of an objective function to maximize subject to a set of constraints. Fig. 7 shows the anatomy of our Integer Linear Programs: The range constraints enforce that $c^{(i)}_{l,m+1} \in \{0, \dots, b-1\}$ and the hard uniformity constraints enforce a non-zero to guarantee the design constraints of stratification, net, and sequence properties as introduced in Sec. 2.2. Remember that matrices M are constructed from the first rows of the C_i matrices and hence include some of the $c^{(i)}_{l,m+1}$. Indicated by $\nu_j = 1$, a satisfied weak constraint adds its weight w_{ν_j} to the objective function. Otherwise, a zero linear combination (stating that corresponding M_j is not full rank) comes along with $\nu_j = 0$.

4 Results

In our previous work [8], we focused on computer graphics applications including image synthesis, parametric texture exploration, and optimal control. However, we only supported the special cases of stratification and $(0, m, s)$-net properties, where $t = 0$. For our new results, we investigate the larger solution space for generator matrices provided by $t \ge 0$ and weak constraints. The latest implementation of the MatBuilder software is publicly available at `https://github.com/loispaulin/matbuilder`.

Fig. 8 shows an initial experiment in $s = 6$ dimensions. We observe that both MatBuilder and LatNetBuilder [5] achieve good performance in terms of discrepancy when maximizing the uniformity. In the following sections, we evaluate more complex profiles to demonstrate the expressiveness of our constraint system.

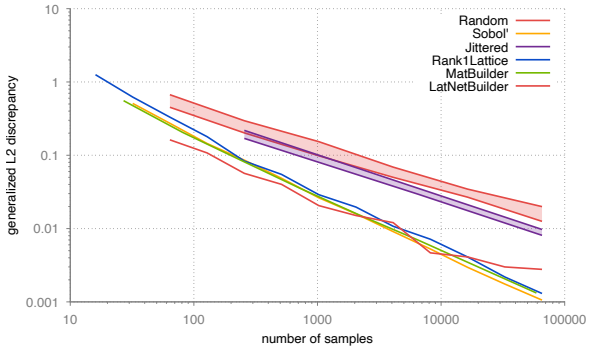


Fig. 8 We measure uniformity in terms generalized l_2-discrepancy in dimension $s = 6$. The MatBuilder results are obtained using a `'weak net t0 0 1 2 3 4 5'` constraint. For LatNetBuilder, we have used a figure of merit minimizing the discrepancy. Rank1Lattice refers to [4].

4.1 Overlapping Net Constraints

In Fig. 9, we present some results of a profile in which we ensure progressive $(0, m, 2)$-net properties for consecutive pairs of dimensions. For all other pairs of dimensions, the additional weak constraints ask the solver to establish a progressive $(0, m, 2)$-net property if possible. Using weak constraints in the profile maximizes the number of elementary intervals checking their part of the progressive $(0, m, 2)$-net property. This leads to points that, even though technically not a progressive $(0, m, 2)$-net, exhibit a similar quality in terms of discrepancy. As compared to the Sobol' sequence, the constraint based generator matrices clearly improve the quality across the 2D sample projections as shown in Fig. 9. Furthermore, when converting these constraints into a loss function for a stochastic matrix construction using LatNetBuilder [5], we observe that it does not obtain comparable sample qualitiy in the projections.

4.2 Playing with t-Values

The constraint system empowers to play with the target t-values. To satisfy a weak constraint, the greedy algorithm maximizes the number of elementary intervals of size b^{-t} containing b^t points, approximating the properties of t-values that are theoretically impossible. For example, in base $b = 3$ the best possible t-value for a progressive $(t, m, 6)$-net is $t = 3$. However, by asking the solver to generate matrices with $t \in \{0, 1, 2\}$ as a weak constraint, we are able to improve uniformity.

In Fig. 10, we evaluate the following generic profiles in $s = 6$ dimensions with increasing t value:

```
b=3
s=6
m=10
weak 1 net t0  0 1 2 3 4 5
```
```
b=3
s=6
m=10
weak 1 net t1  0 1 2 3 4 5
```
```
b=3
s=6
m=10
weak 1 net t2  0 1 2 3 4 5
```
```
b=3
s=6
m=10
weak 1 net t3  0 1 2 3 4 5
```
```
b=3
s=6
m=10
weak 1 net t4  0 1 2 3 4 5
```

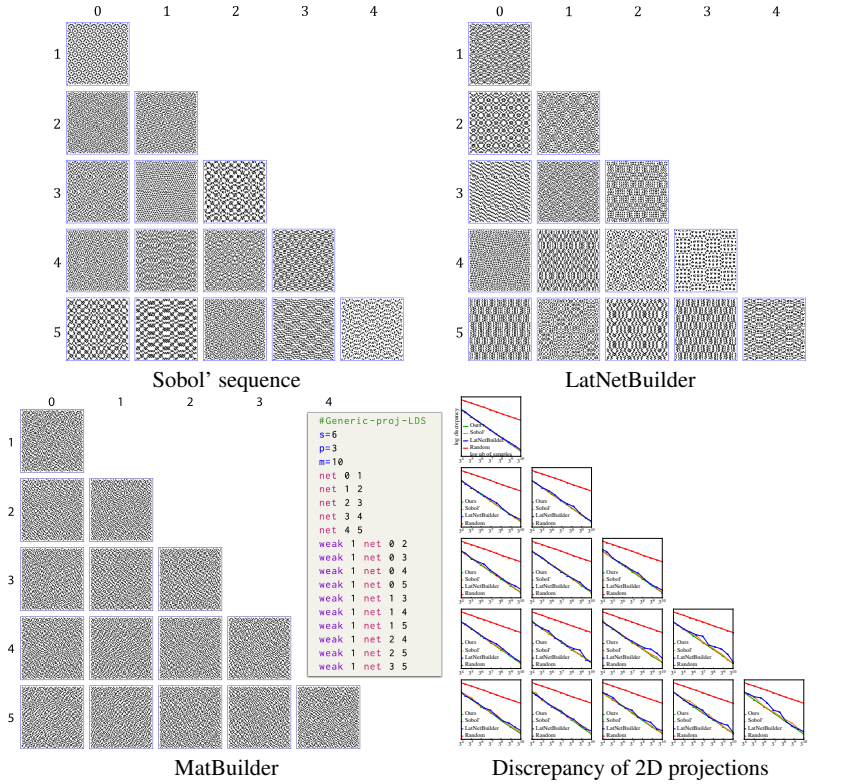
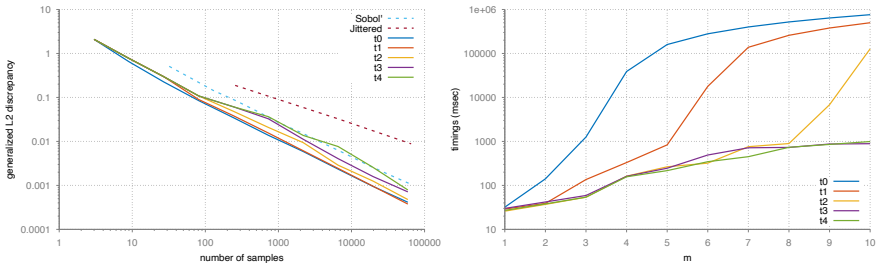


Fig. 9 Comparison of 2D projections. The Sobol' sequence and the result from LatNetBuilder show 2048 samples each, generated in base 2. For MatBuilder, 2187 samples generated in base 3 are shown. The graphs show the discrepancy of the points in comparison. The constraint-based generated points consistently have an excellent low discrepancy.

As expected, increasing the t value has a negative impact on the 6D generalized l_2-discrepancy (Fig. 10-a) and the sample projection uniformity (Fig. 11). In terms of timings of the matrix construction when increasing the m value, n_c as the number of constraints decreases with the t parameter. Hence, the lower is t, the more greedy expansion steps of the matrix columns and rows as m increases (Fig. 10-b).

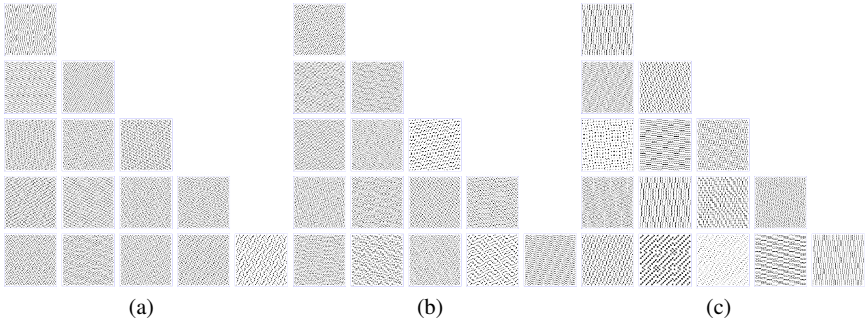
Weighted weak constraints also enable us to negatively weigh a net constraint. While this has little practical purpose, it allows one to explore the range of possible net configurations. We revisit the example of sequences in 6 dimensions in base 3 where the smallest feasible t-value is 3, with the profiles given in Fig. 12.

One can understand such profiles as for instance "What is the best $t = 3$ point set that is not $t = 0$, 1 and 2?". In Figs. 13 and 14, we observe that block artifacts on the projections have a direct impact on the generalized l_2-discrepancy (with relatively similar timings for the matrix construction). However, Fig. 14 shows that the $(3, 6)$-sequence property does not differentiate between points of high or low quality.



(a) (b)

Fig. 10 Performance evaluation when increasing the t parameter from 0 to 4 on a weak net profile in base 3 and dimension 6 up to 3^{10} samples: (a) Quality evaluation in terms of generalized l_2-discrepancy. (b) Timings of the solver as a function of the matrix size m.



(a) (b) (c)

(d) (e)

Fig. 11 Two-dimensional projections of 3^7 samples in dimension 6 following `'weak 1 net ti 0 1 2 3 4 5'` profiles for `i` in $\{0, 1, 2, 3, 4\}$ (from (a) to (e)). Note that dimension indices follow the ones depicted in Fig.9.

5 Conclusion

We extend the MatBuilder software to quality parameters $t > 0$. This enables us to exemplify that the (t, m, s)-net property alone fails to characterize optimal quality

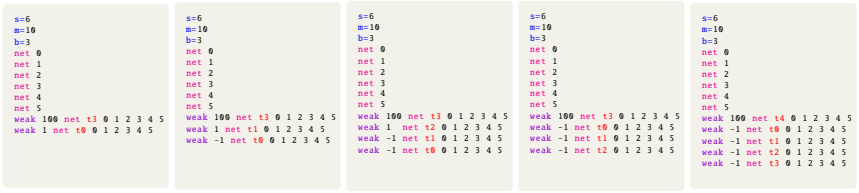
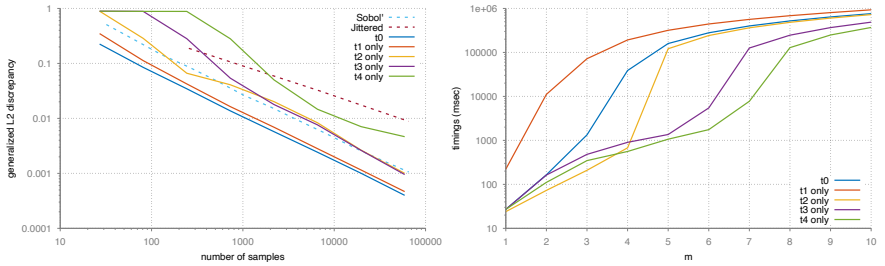


Fig. 12 MatBuilder profiles exploring negatively weighted t constraints: While enforcing the progressive net to be as $t = 3$ as possible (and $t = 4$ for the last one), we progressively invalidate some lower t properties.



(a) (b)

Fig. 13 Performance evaluation of the profiles given in Fig. 12: (a) Quality evaluation in terms of generalized l_2-discrepancy. (b) Timings of the solver as a function of the matrix size m.

both across projections as well as across all dimensions before reaching asymptotic behavior. Using MatBuilder, we are confident that by exploring constraints on not necessarily disjoint subsets of dimensions, partially satisfying constraints, and higher bases, generator matrices can be found that outperform the classic constructions in practice.

References

1. Hong, H.S.: Digital Nets and Sequences for Quasi-Monte Carlo Methods. Ph.D. thesis, Hong Kong Baptist University, China (2002)
2. IBM: IBM ILOG CPLEX Optimizer (2022). URL `https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer`
3. Joe, S., Kuo, F.: Constructing Sobol' sequences with better two-dimensional projections. SIAM J. on Scientific Computing **30**(5), 2635–2654 (2008)
4. Keller, A.: Stratification by rank-1 lattices. In: H. Niederreiter (ed.) Monte Carlo and Quasi-Monte Carlo Methods 2002, pp. 299–313. Springer (2004)
5. L'Ecuyer, P., Munger, D.: Lattice Builder: A general software tool for constructing rank-1 lattice rules. `https://github.com/umontreal-simul/latnetbuilder` (2016)

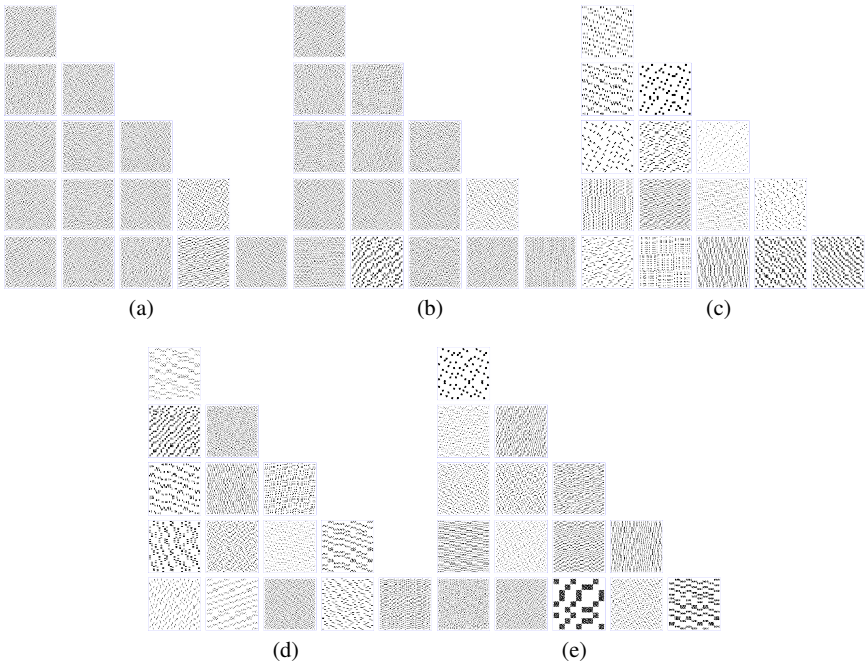


Fig. 14 Two-dimensional projections of 3^7 samples in dimension 6 resulting from `'weak 1 net ti 0 1 2 3 4 5'` profiles for `ti` in $\{0, 1, 2, 3, 4\}$ with negatively weighted `'weak -1 net tj 0 1 2 3 4 5'` constraints for $j < i$ (from (a) to (e)).

6. L'Ecuyer, P., Marion, P., Godin, M., Puchhammer, F.: A tool for custom construction of QMC and RQMC point sets. In: International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, pp. 51–70. Springer (2022)
7. Niederreiter, H.: Random Number Generation and quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA (1992)
8. Paulin, L., Bonneel, N., Coeurjolly, D., Iehl, J.C., Keller, A., Ostromoukhov, V.: MatBuilder: Mastering sampling uniformity over projections. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **41**(4) (2022). DOI https://doi.org/10.1145/3528223.3530063. URL `https://github.com/loispaulin/matbuilder`
9. Sobol', I.M.: On the distribution of points in a cube and the approximate evaluation of integrals. Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki **7**(4), 784–802 (1967)