Efficient usage and construction of QMC methods

Talk at the 13th International Conference on Monte Carlo and Quasi-Monte Carlo Methods (MCQMC 2018)

Adrian Ebert, KU Leuven
Numerical Analysis and Applied Mathematics Section

Joint work with Dirk Nuyens and Peter Kritzer

July 2, 2018

Some point sets ...

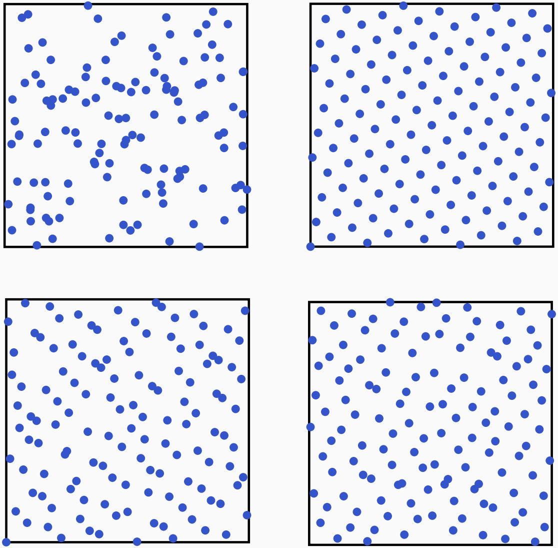


Figure 1: A collection of point sets in $[0,1]^s$

Table of contents

Introduction

Goal of numerical integration

Approximate the integral of the s-variate function f

$$\mathcal{I}(f) := \int_{[0,1]^s} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$$

over the s-dimensional unit cube by a quadrature rule, i.e.,

$$\int_{[0,1]^s} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx \sum_{i=0}^{N-1} w_i f(\boldsymbol{x}_i)\,,$$

with points $\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\} \subseteq [0,1]^s$ and weights $w_0, \ldots, w_{N-1} \in [0,1]$.

Quasi-Monte Carlo method

A *quasi-Monte Carlo (QMC) method* is a quadrature rule

$$Q_{N,s}(f) = \frac{1}{N}\sum_{i=0}^{N-1} f(\boldsymbol{x}_i)\,,$$

with deterministic quadrature points $\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\} \subseteq [0,1]^s$.

Worst-case integration error

Quality measure of choice:

Worst-case error

Let $Q_{N,s}$ be a quasi-Monte Carlo rule with underlying point-set $P = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\} \subseteq [0,1]^s$ and $(\mathcal{H}, \|\cdot\|_{\mathcal{H}})$ be a normed function space. The *worst-case error* of $Q_{N,s}$ w.r.t. $\mathcal{H}$ is defined as

$$e_{N,s}(Q_{N,s}, \mathcal{H}) = \sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \int_{[0,1]^s} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} - \frac{1}{N} \sum_{i=0}^{N-1} f(\boldsymbol{x}_i) \right| .$$

Problem: The quantity $e_{N,s}(Q_{N,s}, \mathcal{H})$ is hard to compute (includes supremum over the unit ball $B = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq 1\}$).

Reproducing kernel Hilbert spaces

Special function space class:

Reproducing kernel Hilbert space (RKHS)

Let $\mathcal{H}$ be a Hilbert space of real-valued functions $f : [0,1]^s \to \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Then $\mathcal{H}$ is called a *reproducing kernel Hilbert space* if there exists a kernel $K : [0,1]^s \times [0,1]^s \to \mathbb{R}$ s.t.

- $K(\cdot, \boldsymbol{x}) \in \mathcal{H}$ for all $\boldsymbol{x} \in [0,1]^s$,
- $f(\boldsymbol{x}) = \langle f, K(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$ and for all $\boldsymbol{x} \in [0,1]^s$.

Here, we consider integrands f belonging to some RKHS $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$.

Examples for RKHS:

- Korobov space of smoothness $\alpha > 1$
- Sobolev spaces of dominating mixed smoothness α
- Walsh function spaces

Worst-case error expression in RKHS

Explicit formula for the squared worst-case error:

Theorem (worst-case error expression)

Let $\mathcal{H}(K)$ be a reproducing kernel Hilbert space with kernel function $K : [0,1]^s \times [0,1]^s \to \mathbb{R}$ such that the integration functional $\mathcal{I}(f)$ is continuous. Then the squared worst-case error of a quasi-Monte Carlo rule $Q_{N,s}$ with quadrature points $\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\}$ takes the form

$$e^2_{N,s}(Q_{N,s}, \mathcal{H}) = \int_{[0,1]^{2s}} K(\boldsymbol{x}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y} - \frac{2}{N} \sum_{i=0}^{N-1} \int_{[0,1]^s} K(\boldsymbol{x}_i, \boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \, .$$

Weighted function spaces

Idea of weighted RKHS:

- based on the concept of effective dimension for high dimensions
- coordinate directions x_j of a function $f : [0,1]^s \to \mathbb{R}$ may have a varying importance w.r.t. their impact on the function value $f(\boldsymbol{x})$
- quantify this importance by assigning a positive number $\gamma_{\mathfrak{u}}$ to each group of variables $\boldsymbol{x}_{\mathfrak{u}}$ which reflects their importance

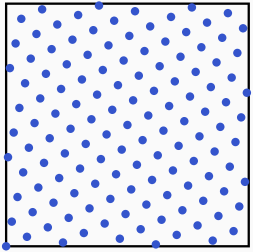
Introduce weighted function spaces[1] with incorporated weight sequence $\boldsymbol{\gamma} = (\gamma_{\mathfrak{u}})_{\mathfrak{u} \in \{1:s\}}$ for each subset $\mathfrak{u} \subseteq \{1, \ldots, s\} =: \{1 : s\}$

Common weight types:

- product weights: $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$ with weights $\gamma_1, \ldots, \gamma_s$
- order-dependent weights: $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|}$ with weights $\Gamma_1, \ldots, \Gamma_s$
- POD weights: $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j$ being a combination of the two

[1] See Sloan, Woźniakowski (1998)

Lattice rules in weighted RKHS



Rank-one lattice rule

A *rank-one lattice rule* is a quasi-Monte Carlo rule with underlying point set $P_N \subseteq [0,1]^s$ of the form

$$P_N = \left\{ \frac{k\,\boldsymbol{z} \bmod N}{N} \;:\; 0 \le k < N \right\} \subseteq [0,1]^s\,,$$

where $\boldsymbol{z} \in \mathbb{Z}^s$ is called the generating vector of the lattice rule.

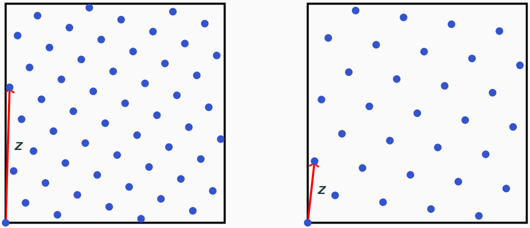


Figure 2: Fibonacci lattice with $N = 55$ and $\boldsymbol{z} = (1, 34)$ (left) and a rank-one lattice with $N = 32$ and $\boldsymbol{z} = (1, 9)$ constructed by the CBC construction (right)

Function space setting

Let $\mathbb{Z}_* := \mathbb{Z} \setminus \{0\}$ and define $r_\alpha(h) := 1/|h|^\alpha$ and $r_\alpha(\boldsymbol{h}) := \prod_{j=1}^{s} r_\alpha(h_j)$ for $h \in \mathbb{Z}_*$ and $\boldsymbol{h} = (h_1, \ldots, h_s) \in \mathbb{Z}_*^s$, respectively.

Consider weighted Korobov space $\mathcal{H}(K_{s,\alpha,\boldsymbol{\gamma}})$ which is RKHS with kernel

$$K_{s,\alpha,\boldsymbol{\gamma}}(\boldsymbol{x}, \boldsymbol{y}) = 1 + \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}} \sum_{\boldsymbol{h}_{\mathfrak{u}} \in \mathbb{Z}_*^{|\mathfrak{u}|}} r_\alpha(\boldsymbol{h}_{\mathfrak{u}}) \exp(2\pi i \boldsymbol{h}_{\mathfrak{u}} \cdot (\boldsymbol{x}_{\mathfrak{u}} - \boldsymbol{y}_{\mathfrak{u}})),$$

with smoothness $\alpha > 1$ and corresponding inner-product

$$\langle f, g \rangle_{K_{s,\alpha,\boldsymbol{\gamma}}} = \sum_{\mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{-1} \sum_{\boldsymbol{h}_{\mathfrak{u}} \in \mathbb{Z}_*^{|\mathfrak{u}|}} r_\alpha^{-1}(\boldsymbol{h}_{\mathfrak{u}}) \, \widehat{f}(\boldsymbol{h}_{\mathfrak{u}}) \overline{\widehat{g}(\boldsymbol{h}_{\mathfrak{u}})}$$

with $\widehat{f}(\boldsymbol{h}_{\mathfrak{u}})$ being the $\boldsymbol{h}_{\mathfrak{u}}$-th Fourier coefficient of f given by

$$\widehat{f}(\boldsymbol{h}_{\mathfrak{u}}) = \int_{[0,1]^s} f(\boldsymbol{x}) \exp(-2\pi i \boldsymbol{h}_{\mathfrak{u}} \cdot \boldsymbol{x}_{\mathfrak{u}}) \, \mathrm{d}\boldsymbol{x}.$$

Connection to the worst-case error

Let $\boldsymbol{z} \in \mathbb{Z}^s$ be the generating vector of a rank-1 lattice rule $\Lambda_{N,s}$ in the weighted Korobov space $\mathcal{H}(K_{s,\alpha,\boldsymbol{\gamma}})$. The squared worst-case error reads

$$e_{N,s}^2(\boldsymbol{z}) = \sum_{\emptyset \neq \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}} \sum_{\boldsymbol{h}_{\mathfrak{u}} \in \mathcal{D}_{\mathfrak{u}}} r_\alpha(\boldsymbol{h}_{\mathfrak{u}})$$

where

$$\mathcal{D}_{\mathfrak{u}} := \left\{ \boldsymbol{h}_{\mathfrak{u}} \in \mathbb{Z}_*^{|\mathfrak{u}|} \ : \ \boldsymbol{h}_{\mathfrak{u}} \cdot \boldsymbol{z}_{\mathfrak{u}} \equiv 0 \pmod N \right\}.$$

There are connections between the worst-case error of the Korobov space $\mathcal{H}(K_{s,2,\boldsymbol{\gamma}})$ and the (root mean square) worst-case error for QMC integration in a weighted (anchored or unanchored) Sobolev space $\mathcal{H}_{s,\gamma}^{\text{sob}}$ using randomly shifted lattice rules or using tent-transformed lattice rules.

Component-by-component type constructions

The component-by-component construction

The component-by-component (CBC) algorithm[2] chooses the components z_i of the generating vector $\boldsymbol{z}$ one component at a time, keeping all previously chosen components fixed.

Algorithm 1 Component-by-component algorithm in RKHS

for $d = 1$ **to** s **do**
    for all $z_d \in \mathbb{U}_N$ **do**
        Calculate $e^2_{N,d}(z_1, z_2, \ldots, z_{d-1}, z_d)$
    end for
    $z_d = \underset{z \in \mathbb{U}_N}{\operatorname{argmin}}\ e^2_{N,d}(z_1, z_2, \ldots, z_{d-1}, z)$
end for

Notation:

$$\begin{aligned} \mathbb{U}_N &= \{z \in \{1, \ldots, N-1\} : \gcd(z, N) = 1\} \\ &= \{1, \ldots, N-1\} \text{ for } N \text{ prime} \end{aligned}$$

[2]See Sloan, Reztsov (2001) or Korobov (1959)

The successive coordinate search construction

Based on an initial vector $\boldsymbol{z}^0$, the successive coordinate search (SCS) algorithm[3] iterates through the components z_i keeping all other components of $\boldsymbol{z}$ and the dimension s fixed.

Algorithm 2 Successive coordinate search (SCS) algorithm

Input: $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$
Output: $\boldsymbol{z} \in \mathbb{U}_N^s$

for $j = 1$ **to** s **do**
  for all $z_j \in \mathbb{U}_N$ **do**
    Calculate $e^2_{N,s}(z_1, \ldots, z_{j-1}, z_j, z^0_{j+1}, \ldots, z^0_s)$
  end for
  $z_j = \operatorname*{argmin}_{z \in \mathbb{U}_N} e^2_{N,s}(z_1, \ldots, z_{j-1}, z_j, z^0_{j+1}, \ldots, z^0_s)$
end for

Note that here: $\boldsymbol{z}^0 \in \mathbb{Z}_N^s = \{0, 1, \ldots N-1\}^s$

[3]See E., Leövey, Nuyens (2018) (to appear in MCQMC 2016 proceedings)

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

$\boldsymbol{z}$

The successive coordinate search algorithm:

$\boldsymbol{z}$

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

z

The successive coordinate search algorithm:

z

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

$\boldsymbol{z}$ z_1 z_2

The successive coordinate search algorithm:

$\boldsymbol{z}$ z_1 z_2 z_3 z_4 z_5 z_6 z_7

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

$\boldsymbol{z}$ $\quad z_1 \quad z_2 \quad z_3$

The successive coordinate search algorithm:

$\boldsymbol{z}$ $\quad z_1 \quad z_2 \quad z_3 \quad z_4 \quad z_5 \quad z_6 \quad z_7$

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

$\boldsymbol{z}$ | z_1 | z_2 | z_3 | z_4

The successive coordinate search algorithm:

$\boldsymbol{z}$ | z_1 | z_2 | z_3 | z_4 | z_5 | z_6 | z_7

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

$\boldsymbol{z}$ z_1 z_2 z_3 z_4 z_5

The successive coordinate search algorithm:

$\boldsymbol{z}$ z_1 z_2 z_3 z_4 z_5 z_6 z_7

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

$\boldsymbol{z}$

The successive coordinate search algorithm:

$\boldsymbol{z}$

Comparison between both constructions

Schema for both constructions:

The component-by-component algorithm:

$\boldsymbol{z}$ z_1 z_2 z_3 z_4 z_5 z_6 z_7

The successive coordinate search algorithm:

$\boldsymbol{z}$ z_1 z_2 z_3 z_4 z_5 z_6 z_7

Error convergence behavior

Assumptions:

- $N = b^m$ with b prime, $m \in \mathbb{N}$ and general weights $\boldsymbol{\gamma} = (\gamma_{\mathfrak{u}})_{\mathfrak{u} \subseteq \{1:s\}}$
- $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$ arbitrary initial vector for the SCS algorithm
- $\boldsymbol{z}_{\text{cbc}}$ and $\boldsymbol{z}_{\text{scs}}$ generating vectors constructed by CBC/SCS algorithm

[4] See, e.g., Dick, Kuo, Sloan (2014) - Acta Numerica review article

[5] See E., Leövey, Nuyens (2018) and E., Kritzer (2018) (submitted)

Error convergence behavior

Assumptions:

- $N = b^m$ with b prime, $m \in \mathbb{N}$ and general weights $\boldsymbol{\gamma} = (\gamma_{\mathfrak{u}})_{\mathfrak{u} \subseteq \{1:s\}}$
- $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$ arbitrary initial vector for the SCS algorithm
- $\boldsymbol{z}_{\text{cbc}}$ and $\boldsymbol{z}_{\text{scs}}$ generating vectors constructed by CBC/SCS algorithm

Let $\alpha > 1$, then $\forall \lambda \in (\frac{1}{\alpha}, 1]$ the worst-case error satisfies:

CBC construction[4]

$$e^2_{N,s}(\boldsymbol{z}_{\text{cbc}}) \le \left(\sum_{\emptyset \neq \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^m} \right)^{\frac{1}{\lambda}}$$

SCS algorithm[5]

$$e^2_{N,s}(\boldsymbol{z}_{\text{scs}}) \le \left(\sum_{j=1}^{s} \sum_{j \in \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^m} \right)^{\frac{1}{\lambda}}$$

[4] See, e.g., Dick, Kuo, Sloan (2014) - Acta Numerica review article
[5] See E., Leövey, Nuyens (2018) and E., Kritzer (2018) (submitted)

Error convergence behavior

Assumptions:

- $N = b^m$ with b prime, $m \in \mathbb{N}$ and general weights $\boldsymbol{\gamma} = (\gamma_{\mathfrak{u}})_{\mathfrak{u} \subseteq \{1:s\}}$
- $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$ arbitrary initial vector for the SCS algorithm
- $\boldsymbol{z}_{\text{cbc}}$ and $\boldsymbol{z}_{\text{scs}}$ generating vectors constructed by CBC/SCS algorithm

Let $\alpha > 1$, then $\forall \lambda \in (\frac{1}{\alpha}, 1]$ the worst-case error satisfies:

CBC construction[4]

$$e_{N,s}^2(\boldsymbol{z}_{\text{cbc}}) \le \left(\sum_{\emptyset \neq \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^m} \right)^{\frac{1}{\lambda}}$$

SCS algorithm[5]

$$e_{N,s}^2(\boldsymbol{z}_{\text{scs}}) \le \left(\sum_{j=1}^{s} \sum_{j \in \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^m} \right)^{\frac{1}{\lambda}}$$

For product weights $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$ the error is $\mathcal{O}(N^{-\alpha/2+\delta})$ with constant independent of the dimension s provided that $\sum_{j=1}^{\infty} \gamma_j^{\frac{1}{\alpha - 2\delta}} < \infty$.

[4] See, e.g., Dick, Kuo, Sloan (2014) - Acta Numerica review article
[5] See E., Leövey, Nuyens (2018) and E., Kritzer (2018) (submitted)

Computational complexity

The CBC and the SCS construction can both be implemented in a fast way[6] by exploiting the (block-)circulant structure of the matrix

$$\Omega_N := \left[\omega\left(\frac{kz \bmod N}{N}\right)\right]_{\substack{z\in\mathbb{U}_N\\ k=0,\ldots,N-1}}$$

to perform a fast matrix-vector multiplication using FFT.

Complexity for $\gamma_{\mathfrak{u}}$ of the form $\prod_{j\in\mathfrak{u}}\gamma_j$

- The fast versions of the algorithms allow for the construction of generating vectors using $\mathcal{O}(s\,N\,\log(N))$ operations.
- The computation in the SCS algorithm is slightly more expensive than in the CBC algorithm since the involved quantities have to be initialized and updated using $\boldsymbol{z}^0$.

[6] See Fast CBC construction by Nuyens and Cools (2006)

Construction method for the SCS algorithm

We consider the following construction methods in order to find generating vectors $\boldsymbol{z}$ with a small worst-case error $e_{N,s}(\boldsymbol{z})$:

1. Uniform random vectors + SCS algorithm:
Choose q initial vectors $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$ at random, apply the fast SCS algorithm to them and then select the one that minimizes $e_{N,s}(\boldsymbol{z})$.
<u>Computational cost (product weights):</u> $\mathcal{O}(q\, s\, N \log(N))$

2. Korobov-type generating vector + SCS algorithm:

Choose q Korobov-type* generating vectors as initial vectors $\boldsymbol{z}^0$, apply the fast SCS algorithm to them and then select the one that minimizes $e_{N,s}(\boldsymbol{z})$.
<u>Computational cost (product weights):</u> $\mathcal{O}(q\, s\, N \log(N))$

*Korobov construction: For a generator $a \in \mathbb{Z}_n$, define the corresponding generating vector by $\boldsymbol{z} = \boldsymbol{z}(a) := (1, a, a^2, \ldots, a^{s-1}) \mod N$.

Reduced CBC type constructions

The reduced construction

Assumptions:

- $N = b^m$ with b prime, $m \in \mathbb{N}$ and general weights $\boldsymbol{\gamma} = (\gamma_{\mathfrak{u}})_{\mathfrak{u} \subseteq \{1:s\}}$
- $w_1, \ldots, w_s \in \mathbb{N}_0$ with $w_1 \leq w_2 \leq \ldots \leq w_s$ and $Y_j = b^{w_j}$ for all j

$$\mathcal{Z}_{N,w_j} := \begin{cases} \{z \in \{1, 2, \ldots, b^{m-w_j} - 1\} : \gcd(z, N) = 1\} & \text{if } w_j < m \\ \{1\} & \text{if } w_j \geq m \end{cases}$$

[7] See Dick, Kritzer, Leobacher, Pillichshammer (2015)

[8] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

The reduced construction

Assumptions:

- $N = b^m$ with b prime, $m \in \mathbb{N}$ and general weights $\boldsymbol{\gamma} = (\gamma_{\mathfrak{u}})_{\mathfrak{u} \subseteq \{1:s\}}$
- $w_1, \ldots, w_s \in \mathbb{N}_0$ with $w_1 \le w_2 \le \ldots \le w_s$ and $Y_j = b^{w_j}$ for all j

$$\mathcal{Z}_{N,w_j} := \begin{cases} \{z \in \{1,2,\ldots,b^{m-w_j}-1\} : \gcd(z,N) = 1\} & \text{if } w_j < m \\ \{1\} & \text{if } w_j \ge m \end{cases}$$

Reduced CBC construction[7]

- Set $z_1 = 1$.
- **For $d = 2$ to s:**

  ◇ Assume that $z_1, \ldots, z_{d-1}$ have already been found.

  ◇ Choose $z_d \in \mathcal{Z}_{N,w_d}$ such that

  $$e^2_{N,d}((Y_1 z_1, \ldots, Y_{d-1} z_{d-1}, Y_d z_d))$$

  is minimized as a function of z_d.
- Obtain the generating vector $\boldsymbol{z} = (Y_1 z_1, \ldots, Y_s z_s)$.

Reduced SCS algorithm[8]

- **Input:** Starting vector $\boldsymbol{z}^0$ with $\boldsymbol{z}^0 = (z_1^0, \ldots, z_s^0) \in \mathbb{Z}_N^s$.
- **For $j = 1$ to s:**

  ◇ Assume that $z_1, \ldots, z_{j-1}$ have already been selected.

  ◇ Choose $z_j \in \mathcal{Z}_{N,w_j}$ such that

  $$e^2_{N,s}((Y_1 z_1, \ldots, Y_{j-1} z_{j-1}, Y_j z_j, z^0_{j+1:s}))$$

  is minimized as a function of z_j.
- Obtain the generating vector $\boldsymbol{z} = (Y_1 z_1, \ldots, Y_s z_s)$.

[7] See Dick, Kritzer, Leobacher, Pillichshammer (2015)

[8] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Error convergence behavior

Let $\boldsymbol{z}_{\text{cbc}} = (Y_1 z_1, \ldots, Y_s z_s)$ and $\boldsymbol{z}_{\text{scs}} = (Y_1 \bar{z}_1, \ldots, Y_s \bar{z}_s)$ be constructed by the reduced CBC or reduced SCS algorithm with arbitrary initial vector $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$, respectively. Let $\alpha > 1$, then $\forall \lambda \in (\frac{1}{\alpha}, 1]$ the worst-case error satisfies:

[9] See Dick, Kritzer, Leobacher, Pillichshammer (2015)
[10] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Error convergence behavior

Let $\boldsymbol{z}_{\text{cbc}} = (Y_1 z_1, \ldots, Y_s z_s)$ and $\boldsymbol{z}_{\text{scs}} = (Y_1 \bar{z}_1, \ldots, Y_s \bar{z}_s)$ be constructed by the reduced CBC or reduced SCS algorithm with arbitrary initial vector $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$, respectively. Let $\alpha > 1$, then $\forall \lambda \in (\frac{1}{\alpha}, 1]$ the worst-case error satisfies:

Reduced CBC construction[9]

$$e_{N,s}^2(\boldsymbol{z}_{\text{cbc}}) \leq \left(\sum_{\emptyset \neq \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^{\max(0, m - \max_{j \in \mathfrak{u}} w_j)}} \right)^{\frac{1}{\lambda}}$$

Reduced SCS algorithm[10]

$$e_{N,s}^2(\boldsymbol{z}_{\text{scs}}) \leq \left(\sum_{j=1}^{s} \sum_{j \in \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^{\max(0, m - w_j)}} \right)^{\frac{1}{\lambda}}$$

[9] See Dick, Kritzer, Leobacher, Pillichshammer (2015)
[10] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Error convergence behavior

Let $\boldsymbol{z}_{\text{cbc}} = (Y_1 z_1, \ldots, Y_s z_s)$ and $\boldsymbol{z}_{\text{scs}} = (Y_1 \bar{z}_1, \ldots, Y_s \bar{z}_s)$ be constructed by the reduced CBC or reduced SCS algorithm with arbitrary initial vector $\boldsymbol{z}^0 \in \mathbb{Z}_N^s$, respectively. Let $\alpha > 1$, then $\forall \lambda \in (\frac{1}{\alpha}, 1]$ the worst-case error satisfies:

Reduced CBC construction[9]

$$e_{N,s}^2(\boldsymbol{z}_{\text{cbc}}) \leq \left(\sum_{\emptyset \neq \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^{\max(0, m - \max_{j \in \mathfrak{u}} w_j)}} \right)^{\frac{1}{\lambda}}$$

Reduced SCS algorithm[10]

$$e_{N,s}^2(\boldsymbol{z}_{\text{scs}}) \leq \left(\sum_{j=1}^{s} \sum_{j \in \mathfrak{u} \subseteq \{1:s\}} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(\alpha\lambda))^{|\mathfrak{u}|}}{b^{\max(0, m - w_j)}} \right)^{\frac{1}{\lambda}}$$

For product weights $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$ the error is $\mathcal{O}(N^{-\alpha/2+\delta})$ with constant independent of the dimension s provided that

$$\sum_{j=1}^{\infty} \gamma_j^{\frac{1}{\alpha - 2\delta}} b^{w_j} < \infty.$$

[9] See Dick, Kritzer, Leobacher, Pillichshammer (2015)
[10] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Computational complexity of the reduced constructions

As before, the reduced CBC and SCS constructions can be implemented in a fast way in the spirit of Nuyens and Cools. This is based on the fact that the (block-)circulant matrix

$$\Omega_{b^m} := \left[\omega\left(\frac{kz \bmod N}{N}\right)\right]_{\substack{z\in\mathbb{U}_N \\ k=0,\dots,N-1}}$$

maintains its (block-)circulant structure upon the substitution $\bar{z} = b^w z$ with $z \in \mathcal{Z}_{N,w}$, i.e.,

$$\Omega_{b^m,w} := \left[\omega\left(\frac{kb^w z \bmod N}{N}\right)\right]_{\substack{z\in\mathcal{Z}_{N,w} \\ k=0,\dots,N-1}}$$

with potentially smaller circulant blocks.

Computational complexity of the reduced constructions

Assumptions:

- $N = b^m$ with b prime, $w_1, \ldots, w_s \in \mathbb{N}_0$ with $w_1 \leq w_2 \leq \ldots \leq w_s$
- s^* is the largest integer such that $w_{s^*} < m$

Reduced CBC construction[11] | **Reduced SCS algorithm**[12]

Product weights $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$

[11] See Dick, Kritzer, Leobacher, Pillichshammer (2015)

[12] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Computational complexity of the reduced constructions

Assumptions:

- $N = b^m$ with b prime, $w_1, \dots, w_s \in \mathbb{N}_0$ with $w_1 \le w_2 \le \dots \le w_s$
- s^* is the largest integer such that $w_{s^*} < m$

Reduced CBC construction[11] | **Reduced SCS algorithm**[12]

$\boxed{\textbf{Product weights } \gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j}$

$$\mathcal{O}\left(N \log N + \min(s, s^*) N + \sum_{j=1}^{\min(s,s^*)} (m - w_j) b^{m - w_j}\right)$$

[11] See Dick, Kritzer, Leobacher, Pillichshammer (2015)

[12] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Computational complexity of the reduced constructions

Assumptions:

- $N = b^m$ with b prime, $w_1, \ldots, w_s \in \mathbb{N}_0$ with $w_1 \leq w_2 \leq \ldots \leq w_s$
- s^* is the largest integer such that $w_{s^*} < m$

Reduced CBC construction[11] | **Reduced SCS algorithm**[12]

$$\boxed{\textbf{Product weights } \gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j}$$

$$\mathcal{O}\left(N \log N + \min(s, s^*) N + \sum_{j=1}^{\min(s,s^*)} (m - w_j) b^{m - w_j} \right)$$

Unreduced constructions: $\mathcal{O}\left(s \, N \log N\right)$

[11] See Dick, Kritzer, Leobacher, Pillichshammer (2015)
[12] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Computational complexity of the reduced constructions

Assumptions:

- $N = b^m$ with b prime, $w_1, \ldots, w_s \in \mathbb{N}_0$ with $w_1 \le w_2 \le \ldots \le w_s$
- s^* is the largest integer such that $w_{s^*} < m$

Reduced CBC construction[11] | **Reduced SCS algorithm**[12]

$$\boxed{\textbf{Product weights } \gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j}$$

$$\mathcal{O}\left(N \log N + \min(s, s^*) N + \sum_{j=1}^{\min(s,s^*)} (m - w_j) b^{m - w_j}\right)$$

Unreduced constructions: $\mathcal{O}\left(s\, N \log N\right)$

$$\boxed{\textbf{POD weights } \gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j}$$

[11] See Dick, Kritzer, Leobacher, Pillichshammer (2015)

[12] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Computational complexity of the reduced constructions

Assumptions:

- $N = b^m$ with b prime, $w_1, \dots, w_s \in \mathbb{N}_0$ with $w_1 \le w_2 \le \dots \le w_s$
- s^* is the largest integer such that $w_{s^*} < m$

Reduced CBC construction[11] | **Reduced SCS algorithm**[12]

Product weights $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$

$$\mathcal{O}\left(N \log N + \min(s, s^*) N + \sum_{j=1}^{\min(s,s^*)} (m - w_j) b^{m - w_j} \right)$$

Unreduced constructions: $\mathcal{O}\left(s\, N \log N\right)$

POD weights $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j$

$$\mathcal{O}\left(N \log N + \min(s, s^*)^2 N + \sum_{j=1}^{\min(s,s^*)} (m - w_j) b^{m - w_j} \right)$$

[11] See Dick, Kritzer, Leobacher, Pillichshammer (2015)

[12] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Computational complexity of the reduced constructions

Assumptions:

- $N = b^m$ with b prime, $w_1, \ldots, w_s \in \mathbb{N}_0$ with $w_1 \leq w_2 \leq \ldots \leq w_s$
- s^* is the largest integer such that $w_{s^*} < m$

Reduced CBC construction[11] | **Reduced SCS algorithm**[12]

$$\boxed{\textbf{Product weights } \gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j}$$

$$\mathcal{O}\left(N \log N + \min(s, s^*) N + \sum_{j=1}^{\min(s,s^*)} (m - w_j) b^{m - w_j}\right)$$

Unreduced constructions: $\mathcal{O}\left(s\, N \log N\right)$

$$\boxed{\textbf{POD weights } \gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j}$$

$$\mathcal{O}\left(N \log N + \min(s, s^*)^2 N + \sum_{j=1}^{\min(s,s^*)} (m - w_j) b^{m - w_j}\right)$$

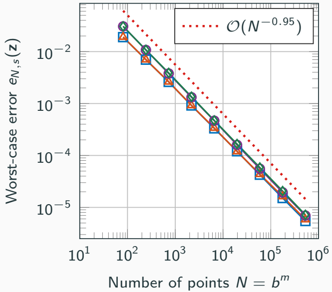
Unreduced construction: $\mathcal{O}(s\, N \log N + s^2 N)$

[11] See Dick, Kritzer, Leobacher, Pillichshammer (2015)

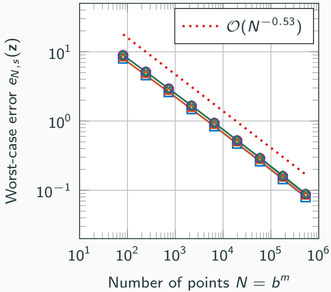
[12] See E., Kritzer (2018) - (submitted for publication - available on arXiv)

Numerical results for product and POD weights

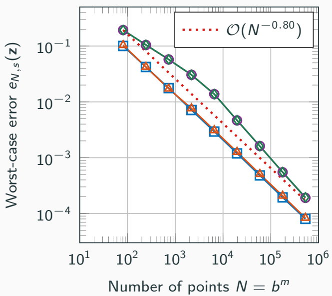
Error convergence for Korobov space with $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$, $s = 100$, $\alpha = 2$, $b = 3$, $w_j = \lfloor 2 \log_b j \rfloor$



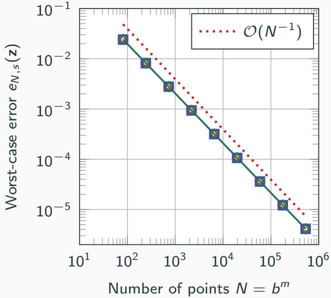
(a) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = (0.2)^j$



(b) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = (0.8)^j$



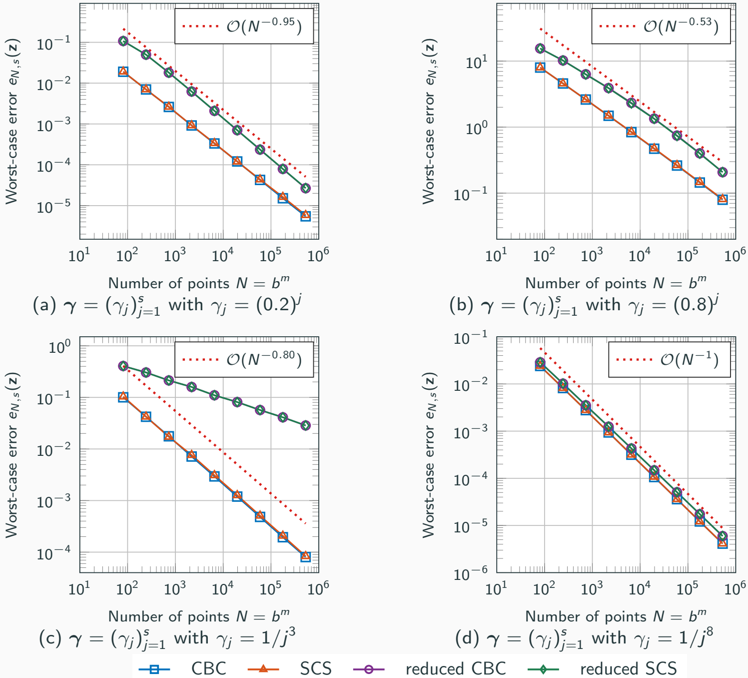
(c) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = 1/j^3$



(d) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = 1/j^8$

CBC SCS reduced CBC reduced SCS

Error convergence for Korobov space with $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}} \gamma_j$, $s = 100$, $\alpha = 2$, $b = 3$, $w_j = \lfloor \frac{7}{2} \log_b j \rfloor$



(a) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = (0.2)^j$

(b) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = (0.8)^j$

(c) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = 1/j^3$

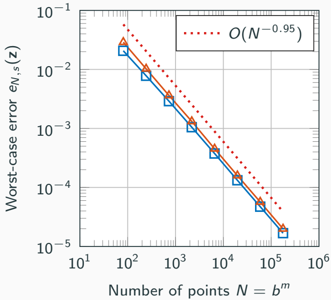
(d) $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^s$ with $\gamma_j = 1/j^8$

Computation times

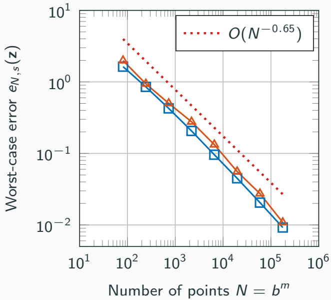
Table 1: Computation times (in seconds) for constructing generating vectors $\boldsymbol{z}$ via the unreduced (normal font) and reduced SCS (**bold font**) algorithm. Constructed for Korobov space with $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}} \gamma_j$, $\alpha = 2$, $b = 2$, $\gamma_j = (0.7)^j$ and $w_j = \lfloor 3 \log_b j \rfloor$.

	$s = 50$	$s = 100$	$s = 500$	$s = 1000$	$s = 2000$
$m = 10$	0.0275 **0.00408**	0.0516 **0.00327**	0.256 **0.00354**	0.516 **0.00347**	1.03 **0.00329**
$m = 12$	0.0418 **0.00592**	0.0751 **0.00504**	0.383 **0.00612**	0.756 **0.00516**	1.56 **0.00794**
$m = 14$	0.0792 **0.014**	0.14 **0.0136**	0.767 **0.0163**	1.39 **0.0138**	2.82 **0.0138**
$m = 16$	0.204 **0.0441**	0.388 **0.0434**	2.09 **0.0434**	4.05 **0.0423**	8.04 **0.0462**
$m = 18$	0.686 **0.16**	1.35 **0.177**	6.89 **0.182**	13.7 **0.183**	26.8 **0.187**
$m = 20$	3.28 **0.843**	6.71 **1.4**	34.4 **1.51**	67.4 **1.37**	132 **1.36**

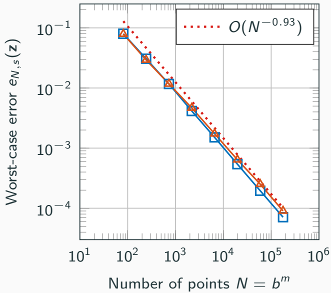
Convergence for Korobov space with $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j$, $s = 500$, $\alpha = 2$, $b = 3$, $w_j = \lfloor 3 \log_b j \rfloor$



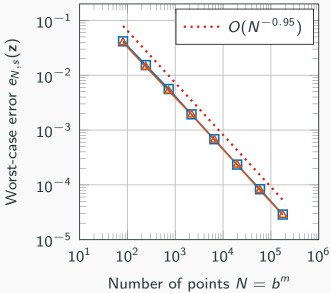
(a) $\Gamma_j = j^4$ and $\gamma_j = (0.1)^j$



(b) $\Gamma_j = j^4$ and $\gamma_j = (0.5)^j$



(c) $\Gamma_j = j^4$ and $\gamma_j = j^{-6}$



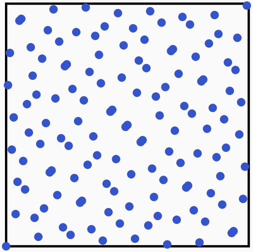
(d) $\Gamma_j = j^4$ and $\gamma_j = j^{-8}$

Computation times

Table 2: Computation times (in seconds) for constructing the generating vector $\boldsymbol{z}$ using the unreduced (normal font) and reduced CBC (**bold font**) construction. The associated lattice can be used for integration in the Korobov space with $\alpha = 2$, $b = 2$ and $w_j = \lfloor 3 \log_b j \rfloor$ with POD weights $\gamma_{\mathfrak{u}} = \Gamma_{|\mathfrak{u}|} \prod_{j \in \mathfrak{u}} \gamma_j$ where $\Gamma_j = j^4, \gamma_j = j^{-6}$.

	$s = 50$	$s = 100$	$s = 200$	$s = 500$	$s = 1000$
$m = 8$	0.136 **0.00393**	0.493 **0.00229**	1.87 **0.00229**	12.1 **0.0021**	44.3 **0.00326**
$m = 10$	0.239 **0.00672**	0.709 **0.00558**	2.72 **0.00654**	16.3 **0.00683**	63 **0.00733**
$m = 12$	0.296 **0.0179**	1.05 **0.0184**	3.91 **0.0167**	24 **0.0227**	95.2 **0.0274**
$m = 14$	0.552 **0.0757**	2.03 **0.0876**	7.49 **0.0932**	45.8 **0.115**	180 **0.157**
$m = 16$	1.62 **0.576**	6.31 **0.583**	24.2 **0.576**	148 **0.631**	589 **0.585**
$m = 18$	5.77 **3.3**	21.8 **4.86**	83.8 **4.85**	512 **4.85**	2013 **5.26**
$m = 20$	27.8 **17.4**	103 **59.9**	401 **61.4**	2452 **58.9**	9791 **61.7**

Digital net constructions



Digital construction scheme

Assumptions:

- b prime, $\mathbb{Z}_b := \mathbb{Z}/b\mathbb{Z} = \{0, 1, \ldots, b-1\}$
- $C_1, \ldots, C_s \in \mathbb{Z}_b^{m \times m}$, i.e., m-by-m generating matrices with entries in $\mathbb{Z}_b$
- $P_N = P_{b^m} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{b^m-1}\}$ with $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,s})$ for $i = 0, \ldots, b^m - 1$

Construction scheme: For each $i = 0, \ldots, b^m - 1$ consider the following steps:

1. Compute the base b expansion of $i = i_1 + i_2 b + \ldots + i_m b^m$.
2. For $j = 1, \ldots, s$ compute $y_1^{(j)}, \ldots, y_m^{(j)}$ (with operations in $\mathbb{Z}_b$) via
$$\begin{pmatrix} y_1^{(j)} \\ y_2^{(j)} \\ \vdots \\ y_m^{(j)} \end{pmatrix} = C_j \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_m \end{pmatrix}.$$
3. Define the component $x_{i,j}$ by
$$x_{i,j} = \sum_{k=1}^{m} y_k^{(j)} b^{-k} = \frac{y_1^{(j)}}{b} + \frac{y_2^{(j)}}{b^2} + \ldots + \frac{y_m^{(j)}}{b^m}.$$

Then $P_{b^m} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{b^m-1}\}$ is called a digital net in base b.

QMC software and generators

1. Software on Dirk Nuyens' webpage to apply quasi-Monte Carlo:
   - The Magic Point Shop!:
     `https://www.cs.kuleuven.be/~dirkn/qmc-generators/`
   - QMC4PDE:
     `https://www.cs.kuleuven.be/~dirkn/qmc4pde/`

2. Git repositories with C++, Python, Matlab code on Bitbucket:
   - `https://bitbucket.org/dnuyens/qmc-generators`
   - `https://bitbucket.org/dnuyens/qmc4pde/`

3. Git repository with software for presented CBC type lattice constructions (Matlab) and digital net generator:
   - `https://bitbucket.org/adrian_ebert/digital_sequences/`

Specification for using the digital net point generator

Command line generator `digitalseq_b2g` with the following inputs:

- s – number of dimensions s
- m – number of points given by $N = 2^m$
- C (via file) – generating matrices $C_1, \ldots, C_s$ given in column format

$$C_j = [c_1^{(j)}, c_2^{(j)}, \ldots, c_m^{(j)}] := \begin{pmatrix} c_{1,1}^{(j)} & c_{2,1}^{(j)} & \cdots & c_{m,1}^{(j)} \\ c_{1,2}^{(j)} & c_{2,2}^{(j)} & \cdots & c_{m,2}^{(j)} \\ \vdots & \vdots & & \vdots \\ c_{1,m}^{(j)} & c_{2,m}^{(j)} & \cdots & c_{m,m}^{(j)} \end{pmatrix}$$

with $c_i^{(j)} = \sum_{k=1}^m c_{i,k}^{(j)} 2^k \in \{0, 1, \ldots, N-1\}$ for $i = 1, \ldots, m$.

Features available with this digital net point generator:

- Generation of digital sequence points based on C
- Digital randomization techniques (see following slide)
- Adaptable state of the sequence generator
- Digital interlacing of factor α (via separate programme)

Digital shifting and scrambling

The digital generator comes with the following two randomization techniques:

1. Digital shift:
   - $\Delta \in [0,1]^s$ with base b expansion $\Delta_j = \sum_{k=1}^{\infty} d_k^{(j)} b^{-k}$
   - $P_{b^m} = \{\boldsymbol{x}_0 \oplus \Delta, \ldots, \boldsymbol{x}_{b^m-1} \oplus \Delta\}$ with component-wise digit-wise addition in base b
2. Linear Matousek scrambling:
   - Consider point $\boldsymbol{x} \in [0,1]^s$ with components $x_j = \sum_{k=1}^{\infty} x_k^{(j)} b^{-k}$
   - Linear matrix scramble: $x_j \to \bar{x}_j = \sum_{k=1}^{\infty} \bar{x}_k^{(j)} b^{-k}$ with

$$\bar{x}_k^{(j)} = \sum_{i=1}^{k} M_{ki} x_i^{(j)}.$$

   - Expressible via generating matrices: Set $\bar{C}_j := M_j \cdot C_j$ with

$$M_j = \begin{pmatrix} u_{1,1} & 0 & \cdots & 0 \\ r_{2,1} & u_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ r_{m,1} & \cdots & r_{m,m-1} & u_{m,m} \end{pmatrix}$$

and uniform random numbers $u_{k,i} \in \mathbb{Z}_b \setminus \{0\}$ and $r_{k,i} \in \mathbb{Z}_b$.

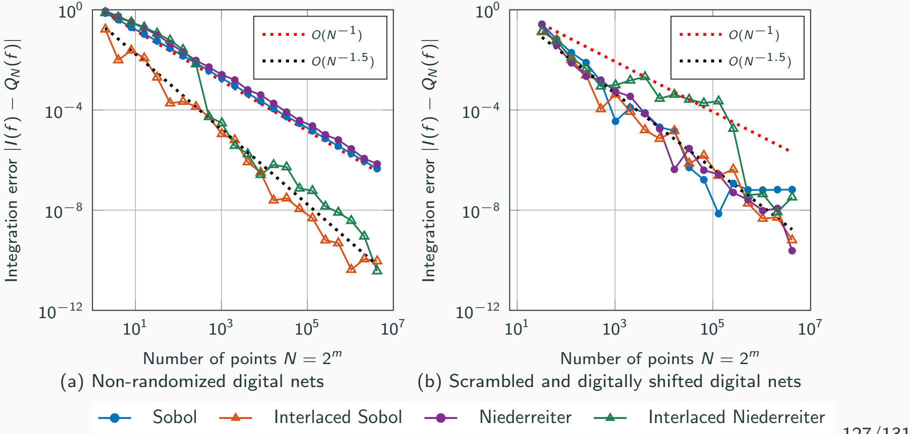
Demo of the digital net point generator

Terminal demo

Numerical example 1

Function $f(x) = \exp(c\sum_{j=1}^{s} \frac{x^j}{j^\beta})$ with integral $I(f) = \prod_{j=1}^{s} \frac{\exp(c\,j^{-\beta})-1}{c\,j^{-\beta}}$.

- $N = 2^m, s = 50, \beta = 2, c = 1$ and we consider digital interlacing of factor $\alpha = 2$.
- We consider $t = 2^4$ affine matrix scrambles and $Q_N(f) := \frac{1}{t}\sum_{i=1}^{t} Q_{2^{m-4}}(f)$.
- In this example, we first interlace and then randomize the digital sequence.



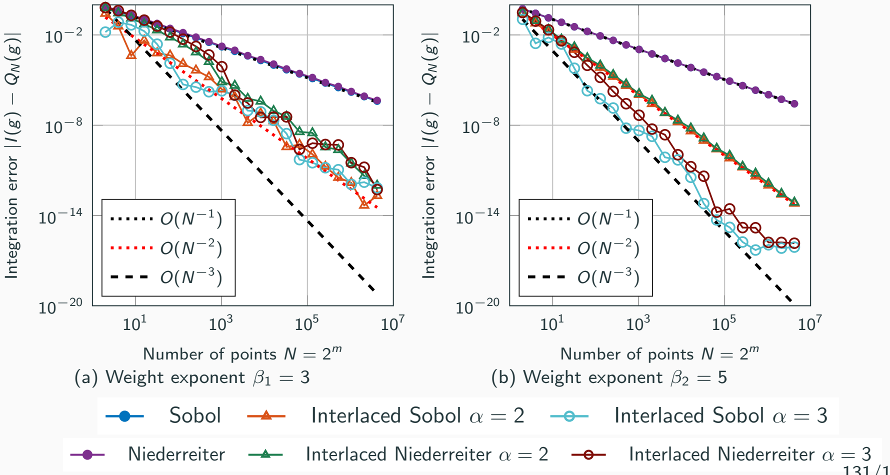
(a) Non-randomized digital nets

(b) Scrambled and digitally shifted digital nets

Numerical example 2

Function $g(\boldsymbol{x}) = \prod_{j=1}^{s} \left(1 + \frac{2x_j - a}{j^\beta}\right)$ with integral $I(g) = \prod_{j=1}^{s} \left(1 + \frac{1-a}{j^\beta}\right)$.

- $N = 2^m$, $s = 100$, $a = 0$ and we consider digital interlacing of factor $\alpha \in \{2, 3\}$.
- We consider two different weight exponents $\beta_1 = 3$ and $\beta_2 = 5$.



(a) Weight exponent $\beta_1 = 3$

(b) Weight exponent $\beta_2 = 5$

Thank you for your attention!

Introduction
Lattice rules
CBC type constructions
Reduced CBC
Numerical results
Digital construction