Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

# Computational Higher Order Quasi-Monte Carlo Integration

R. N. Gantner and Ch. Schwab

Research Report No. 2014-25
September 2014

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Funding SNF: 149819
Funding ERC: AdG247277

# Computational Higher Order Quasi-Monte Carlo Integration

Robert N. Gantner and Christoph Schwab

**Abstract** The efficient construction of higher-order interlaced polynomial lattice rules introduced recently in [4] is considered. After briefly reviewing the principles of their construction by the "fast component-by-component" (CBC) algorithm due to [1, 10] as well as recent theoretical results on their convergence rates, we indicate algorithmic details of their construction. Instances of such rules are applied to high-dimensional test integrands which belong to weighted function spaces with weights of product and of SPOD type. Practical considerations that lead to improved quantitative convergence behavior for various classes of test integrands are reported. The use of (analytic or numerical) bounds on the Walsh coefficients of the integrand are found to improve the convergence behavior. The sharpness of theoretical bounds on memory usage and operation counts, with respect to the number of points $N$ and dimension $s$ of the integration domain is verified experimentally. The efficiency of the proposed algorithms for computation of the generating vectors is confirmed for the considered classes of functions in dimensions $s = 10, ..., 1000$.

## 1 Introduction

The efficient approximation of high-dimensional integrals is a core task in many areas of scientific computing. We mention only uncertainty quantification, computational finance, computational physics and chemistry, and computational biology. More specifically, high-dimensional integrals arise in the computation of statistical quantities of solutions to partial differential equations with random inputs.

In addition to efficient spatial and temporal discretizations of partial differential equation models, it is important to devise high-dimensional quadrature schemes that are able to exploit an implicity lower-dimensional structure in parametric input data

Robert N. Gantner · Christoph Schwab
Seminar for Applied Mathematics, ETH Zürich, Rämistrasse 101, Zürich, Switzerland e-mail: robert.gantner@sam.math.ethz.ch, e-mail: christoph.schwab@sam.math.ethz.ch

and solutions. The rate of convergence of Monte Carlo (MC) methods is *dimension-robust*, i.e. the convergence rate bound holds with constants independent of the problem dimension, but it is limited to $1/2$. Thus it is important to devise dimension-robust methods of higher order.

In recent years, numerous approaches to achieve this type of higher-order convergence have been proposed; we mention only quasi Monte-Carlo integration, adaptive Smolyak quadrature, adaptive polynomial chaos discretizations, and related methods.

In the present paper, we consider the realization of novel higher-order interlaced polynomial lattice rules introduced in [4, 6], which allow an integrand-adapted construction of a quasi-Monte Carlo quadrature rule that exploits sparsity of the parameter-to-solution map. We consider in what follows the problem of integrating a function $f : [0,1)^s \to \mathbb{R}$ of $s$ variables $y_1, \ldots, y_s$ over the $s$-dimensional unit cube,

$$\mathscr{I}[f] := \int_{[0,1)^s} f(y_1, \ldots, y_s)\, \mathrm{d}y_1 \cdots \mathrm{d}y_s. \tag{1}$$

Of course, exact computation quickly becomes infeasible and we must, in most applications, resort to an approximation of (1) by a quadrature rule. We focus on quasi-Monte Carlo quadrature rules; more specifically, we consider interlaced polynomial lattice point sets for functions in weighted spaces with weights of product and SPOD type. Denoting the interlaced polynomial lattice point set by $\mathscr{P} = \{\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(N-1)}\}$ with $\boldsymbol{x}^{(n)} \in [0,1)$ for $n = 0, \ldots, N-1$, we write the QMC quadrature rule as

$$\mathscr{Q}_{\mathscr{P}}[f] := \frac{1}{N} \sum_{n=0}^{N-1} f(\boldsymbol{x}^{(n)}).$$

In section 2 we first define in more detail the structure of the point set $\mathscr{P}$ considered throughout and derive worst-case error bounds for integrand functions which belong to certain weighted spaces of functions introduced in [8]. Then, component-by-component construction is briefly reviewed and the worst-case error reformulated to allow efficient computation. The main contribution of this paper is found in sections 4 and 5, which mention some practical considerations required for efficient implementation and application of these rules. In section 5, we give measured convergence results for example integrands, showing the applicability of these methods.

## 2 Interlaced Polynomial Rank-1 Lattice Rules

Polynomial rank-1 lattice point sets, introduced by Niederreiter in [9], are a modification of standard rank-1 lattice point sets to polynomial arithmetic in $\mathbb{Z}_b[x]$. A *polynomial lattice rule* is an equal-weight quasi-Monte Carlo (QMC) quadrature rule based on such point sets. Here, we consider the higher-order interlaced polynomial lattice rules introduced in [4, Def. 3.6], [3, Def. 5.1] and focus on computational techniques for their efficient construction.

## 2.1 Definitions

For a given prime number $b \geq 2$ and an integer $m \in \mathbb{N}$, we denote by $G_{b,m} = \{p(x) \in \mathbb{Z}_b[x] \backslash \{0\} : \deg(p) < m\}$ the cyclic group formed by the nonzero elements of the finite field $\mathbb{Z}_{b^m}$ under multiplication. Let $P \in \mathbb{Z}_b[x]$ be an irreducible polynomial of degree $m$. Throughout, we frequently interchange an integer $n$, $0 \leq n < N = b^m$, with its associated polynomial $n(x) = \eta_0 + \eta_1 x + \eta_2 x^2 + \ldots + \eta_{m-1} x^{m-1}$, the coefficients of which are given by the $b$-adic expansion $n = \eta_0 + \eta_1 b + \eta_2 b^2 + \ldots + \eta_{m-1} b^{m-1}$.

Given a *generating vector* $\mathbf{q} \in G_{b,m}^s$, we have the following expression for the $i$-th component of the $n$-th point $\mathbf{x}^{(n)} \in [0,1)^s$ of a polynomial lattice point set $\mathscr{P}$:

$$x_i^{(n)} = v_m\Big(\frac{n(x)q_i(x)}{P(x)}\Big), \quad i = 1, \ldots, s,\ n = 0, \ldots, N-1,$$

where the mapping $v_m : \mathbb{Z}_b((x^{-1})) \to [0,1)$ is given by $v_m\left(\sum_{\ell=1}^{\infty} \xi_\ell x^{-\ell}\right) = \sum_{\ell=1}^{m} \xi_\ell b^{-\ell}$.

A key component for obtaining higher-order convergence rates is the *interlacing* of lattice point sets, as introduced in [6]. To this end, we define the digit interlacing function, which maps $\alpha$ points in $[0,1)$ to one point in $[0,1)$.

**Definition 1 (Digit Interlacing Function).** We define the *digit interlacing function* $\mathscr{D}_\alpha$ with *interlacing factor* $\alpha \in \mathbb{N}$ acting on the points $\{x_j \in [0,1), j = 1, \ldots, \alpha\}$ by

$$\mathscr{D}_\alpha(x_1, \ldots, x_\alpha) = \sum_{a=1}^{\infty} \sum_{j=1}^{\alpha} \xi_{j,a} b^{-j-\alpha(a-1)},$$

where by $\xi_{j,a}$ we denote the $a$-th component of the $b$-adic decomposition of $x_j$, $x_j = \xi_{j,1} b^{-1} + \xi_{j,2} b^{-2} + \ldots$.

An interlaced polynomial lattice point set based on the generating vector $\mathbf{q} \in G_{b,m}^{\alpha s}$, which is now $\alpha$ times larger than before, is then given by the points $\{\mathbf{x}^{(n)}\}_{n=0}^{b^m-1}$ with

$$x_i^{(n)} = \mathscr{D}_\alpha\left(v_m\left(\frac{n(x)q_{\alpha(i-1)+1}(x)}{P(x)}\right), \ldots, v_m\left(\frac{n(x)q_{\alpha(i-1)+\alpha}(x)}{P(x)}\right)\right), \quad i = 1, \ldots, s,$$

i.e. the $i$-th coordinate of the $n$-th point is obtained by interlacing a block of $\alpha$ points.

## 2.2 Worst-Case Error Bound

We give here a brief overview of bounds on the worst case error which are required for CBC construction; for details we refer to [4]. The results therein were based on the "new function space setting" from [8], which generalizes the notion of a *reproducing kernel Hilbert space* to a Banach space setting.

### 2.2.1 Function Space Setting

In order to derive a worst-case error bound, consider the *higher-order unanchored Sobolev space* $\mathscr{W}_{s,\alpha,\gamma,q,r} := \{f \in L^1([0,1)^s) : \|f\|_{s,\alpha,\gamma,q,r} < \infty\}$ where the $r$-th power of the norm $\|\cdot\|_{s,\alpha,\gamma,q,r}$ is given by

$$\|f\|^r_{s,\alpha,\gamma,q,r} := \sum_{\mathfrak{u}\subseteq\{1:s\}} \left\| \gamma_{\mathfrak{u}}^{-1} \sum_{\mathfrak{v}\subseteq\mathfrak{u}} \sum_{\tau\in\{1:\alpha\}^{|\mathfrak{u}\setminus\mathfrak{v}|}} \int_{[0,1]^{s-|\mathfrak{v}|}} (\partial_{\boldsymbol{y}}^{(\alpha_{\mathfrak{v}},\tau,0)} f)(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}_{\{1:s\}\setminus\mathfrak{v}} \right\|^r_{L^q}, \quad (2)$$

where $(\alpha_{\mathfrak{v}},\tau,0)$ denotes the sequence $\boldsymbol{\nu}$ with $\nu_j = \alpha$ for $j \in \mathfrak{v}$, $\nu_j = \tau_j$ for $j \in \mathfrak{u}\setminus\mathfrak{v}$ and $\nu_j = 0$ for $j \notin \mathfrak{u}$, and by $\{1:s\}$ we mean $\{1,2,\ldots,s\}$. The space $\mathscr{W}_{s,\alpha,\gamma,q,r}$ consists of smooth functions with integrable mixed derivatives of orders up to $\alpha$ with respect to each variable, and $L^q$-integrable ($q \in [1,\infty]$) mixed derivatives containing a derivative of order $\alpha$ in at least one variable. The norm (2) is defined by combining the $L^q$ norms of the derivatives in the $\ell^r$ sense for $r \in [1,\infty]$. This space is called *unanchored* because the innermost integral over $[0,1]^{s-|\mathfrak{v}|}$ in (2) integrates out the "inactive" coordinates, i.e. those with respect to which a derivative of order less than $\alpha$ is taken, rather than "anchoring" these variables by fixing their values equal to an anchor point $a \in [0,1)^s$.

### 2.2.2 Error Bound

The *worst-case error* $e^{\mathrm{WC}}(\mathscr{P},\mathscr{W})$ of a point set $\mathscr{P} = \{\boldsymbol{y}^{(0)},\ldots,\boldsymbol{y}^{(b^m-1)}\}$ over the function space $\mathscr{W}$ is defined by the following supremum over the unit ball in $\mathscr{W}$:

$$e^{\mathrm{WC}}(\mathscr{P},\mathscr{W}) = \sup_{\|f\|_{\mathscr{W}}\le 1} |\mathscr{I}[f] - \mathscr{Q}_{\mathscr{P}}[f]|.$$

Assume $1/r + 1/r' = 1$, $\alpha, s \in \mathbb{N}$ with the condition $\alpha > 1$ on the interlacing parameter. Define a collection of positive weights $\gamma = (\gamma_{\mathfrak{u}})_{\mathfrak{u}\subset\mathbb{N}}$. Then, by [4, Thm. 3.5], we have the following bound on the worst-case error in the space $\mathscr{W}_{s,\alpha,\gamma,q,r}$,

$$\sup_{\|f\|_{\mathscr{W}_{s,\alpha,\gamma,q,r}}\le 1} |\mathscr{I}[f] - \mathscr{Q}_{\mathscr{P}}[f]| \le e_{s,\alpha,\gamma,r'}(\mathscr{P}),$$

with the bound for the worst case error $e_{s,\alpha,\gamma,r'}(\mathscr{P})$ given by

$$e_{s,\alpha,\gamma,r'}(\mathscr{P}) = \left( \sum_{\emptyset\neq\mathfrak{u}\subseteq\{1:s\}} \left( C_{\alpha,b}^{|\mathfrak{u}|} \gamma_{\mathfrak{u}} \sum_{k_{\mathfrak{u}}\in\mathscr{D}^\star_{\mathfrak{u}}} b^{-\mu_\alpha(k_{\mathfrak{u}})} \right)^{r'} \right)^{1/r'}. \quad (3)$$

The constant $C_{\alpha,b}$ is obtained by bounding the Walsh coefficients of functions in Sobolev spaces, see [2, Thm. 14] for details. Here, it has the value

$$C_{\alpha,b} = \max\left(\frac{2}{(2\sin\frac{\pi}{b})^{\alpha}}, \max_{z=1,\ldots,\alpha-1}\frac{1}{(2\sin\frac{\pi}{b})^{z}}\right) \times \left(1+\frac{1}{b}+\frac{1}{b(b+1)}\right)^{\alpha-2}\left(3+\frac{2}{b}+\frac{2b+1}{b-1}\right). \tag{4}$$

The bound (3) holds for general digital nets; however, we wish to restrict ourselves to polynomial lattice rules. We additionally choose $r' = 1$ (and thus $r = \infty$, i.e. the $\ell^\infty$ norm over the sequence indexed by $\mathfrak{u} \subseteq \{1:s\}$ in the norm (2)). In the following we use $\omega(y) = \frac{b-1}{b^\alpha - b} - b^{\lfloor \log_b y \rfloor(\alpha-1)}\frac{b^\alpha-1}{b^\alpha-b}$ where $\omega(0) = \frac{b-1}{b^\alpha-b}$. Using [8, Thm. 3.9], we rewrite the sum over the dual net $\mathscr{D}_{\mathfrak{u}}^\star$ in (3) in a computationally amenable form,

$$e_{s,\alpha,\gamma,1}(\mathscr{P}) \le E_d(\boldsymbol{q}) = \frac{1}{b^m}\sum_{n=0}^{b^m-1}\sum_{\substack{\mathfrak{v}\subseteq\{1:d\}\\ \mathfrak{v}\neq\emptyset}}\widetilde{\gamma}_{\mathfrak{v}}\prod_{j\in\mathfrak{v}}\omega(y_j^{(n)}), \quad \boldsymbol{y}^{(n)} \in \mathscr{P}, \tag{5}$$

where $y_j^{(n)} = v_m\left(\frac{n(x)q_j(x)}{P(x)}\right)$ depends on the $j$-th component of the generating vector, $q_j(x)$, and the auxiliary weight $\widetilde{\gamma}_{\mathfrak{v}}$ depends on the choice of weights. For product weights, we can define

$$\widetilde{\gamma}_{\mathfrak{v}} = \prod_{j\in\mathfrak{u}(\mathfrak{v})}\gamma_j, \qquad \gamma_j = C_{\alpha,b}b^{\alpha(\alpha-1/2)}\sum_{\nu=1}^{\alpha}\nu!2^{\delta(\nu,\alpha)}\beta_j^{\nu}, \tag{6}$$

and obtain from (5) the worst-case error bound

$$E_d(\boldsymbol{q}) = \frac{1}{b^m}\sum_{n=0}^{b^m-1}\sum_{\substack{\mathfrak{u}\subseteq\{1:s\}\\ \mathfrak{u}\neq\emptyset}}\Big(\prod_{j\in\mathfrak{u}}\gamma_j\Big)\sum_{\substack{\mathfrak{v}\subseteq\{1:d\}\\ \mathfrak{u}(\mathfrak{v})=\mathfrak{u}}}\Big(\prod_{j\in\mathfrak{v}}\omega(y_j^{(n)})\Big). \tag{7}$$

For SPOD weights we have

$$\widetilde{\gamma}_{\mathfrak{v}} = \sum_{\boldsymbol{\nu}_{\mathfrak{u}(\mathfrak{v})}\in\{1:\alpha\}^{|\mathfrak{u}(\mathfrak{v})|}}|\boldsymbol{\nu}_{\mathfrak{u}(\mathfrak{v})}|!\prod_{j\in\mathfrak{u}(\mathfrak{v})}\gamma_j(\nu_j), \quad \gamma_j(\nu_j) = C_{\alpha,b}b^{\alpha(\alpha-1)/2}2^{\delta(\nu_j,\alpha)}\beta_j^{\nu_j}, \tag{8}$$

for which we obtain

$$E_d(\boldsymbol{q}) = \frac{1}{b^m}\sum_{n=0}^{b^m-1}\sum_{\substack{\mathfrak{v}\subseteq\{1:d\}\\ \mathfrak{v}\neq\emptyset}}\sum_{\nu\in\{1:\alpha\}^{|\mathfrak{u}(\mathfrak{v})|}}|\nu|!\Big(\prod_{j\in\mathfrak{u}(\mathfrak{v})}\gamma_j(\nu_j)\Big)\Big(\prod_{j\in\mathfrak{v}}\omega(y_j^{(n)})\Big). \tag{9}$$

These two expressions will be the basis of the component-by-component (CBC) construction elucidated in the next section. We note that the powers of $C_{\alpha,b}$ arising in (7) and (9) can become very large, leading to a pronounced negative impact on the construction procedure (see Section 4.1 below). The constant $C_{\alpha,b}$, defined in (4), stems from bounds on the Walsh coefficients of smooth functions [2].

## 3 Component-by-Component Construction

The component-by-component construction (CBC) [7, 13, 14] is a simple but nevertheless effective algorithm for computing generating vectors for rank-1 lattice rules, of both standard and polynomial type. In each iteration of the algorithm, the worst-case error is computed for all candidate elements of the generating vector, and the one with minimal WCE is taken as the next component. After $s$ iterations, a generating vector of length $s$ is obtained, which can then be used for QMC quadrature.

Nuyens and Cools reformulated in [1, 10] the CBC construction to exploit the cyclic structure inherent in the point sets for standard lattice rules when the dimension $s$ is a prime number. This leads to the so-called *Fast CBC* algorithm based on the fast Fourier transform (FFT) which speeds up the computation drastically. It is also the basis for the present construction.

Fast CBC is based on reformulating (7) and (9): instead of iterating over the index $d = 1,\dots,\alpha s_{max}$, we iterate over the dimension $s = 1,\dots,s_{max}$ and for each $s$ over $t = 1,\dots,\alpha$. Thus, the index $d$ above is replaced by the pair $s,t$ through $d = \alpha(s-1)+t$ and we write

$$y_{i,j}^{(n)} = y_{\alpha(i-1)+j}^{(n)}, \quad i = 1,\dots,s_{max},\ j = 1,\dots,\alpha. \tag{10}$$

In order to obtain an efficient algorithm we further reformulate (7) and (9) such that only intermediate quantities are updated instead of computing $E_d(\boldsymbol{q})$ in (7) and (9) from scratch.

### 3.1 Product Weights

In the product weight case, we have for $t = \alpha$ the expression

$$E_{s,\alpha}(\boldsymbol{q}) = \frac{1}{b^m}\sum_{n=0}^{b^m-1}\prod_{j=1}^{s}\left[1+\gamma_j\left(\prod_{i=1}^{\alpha}(1+\omega(y_{j,i}^{(n)}))-1\right)\right]-1. \tag{11}$$

We define the quantity $Y_s(n) = \prod_{j=1}^{s}\left[1+\gamma_j\left(\prod_{i=1}^{\alpha}(1+\omega(y_{j,i}^{(n)}))-1\right)\right]$ which will be updated at the end of each iteration over $t$. To emphasize the independence of certain quantities on the current unknown (the $s,t$ component of $\boldsymbol{q}$), we denote the truncated generating vector by $\boldsymbol{q}_d = (q_1,\dots,q_d)$ or in analogy to (10), $\boldsymbol{q}_{s,t} = (q_1,\dots,q_{s,t})$. We now want to write $E_{s,t}(\boldsymbol{q}_{s,t}) = E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha}) + \widetilde{E}(q_{s,1},\dots,q_{s,t})$, such that (11) can be used for $E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha})$ during the iteration over $t$. For $t < \alpha$, we have

$$E_{s,t}(\boldsymbol{q}) = \frac{1}{b^m}\sum_{n=0}^{b^m-1}\left[1+\gamma_s\left(\prod_{i=1}^{t}(1+\omega(y_{s,i}^{(n)}))-1\right)\right]Y_{s-1}(n)-1,$$

which can be written in terms of $E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha})$ as

$$E_{s,t}(\boldsymbol{q}) = E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha}) + \frac{\gamma_s}{b^m}\sum_{n=0}^{b^m-1} Y_{s-1}(n) + \frac{\gamma_s}{b^m}\sum_{n=0}^{b^m-1}\left(\prod_{i=1}^{t}(1+\omega(y_{s,i}^{(n)}))\right) Y_{s-1}(n).$$

For later use and ease of exposition, we define $V_{s,t}(n) = \prod_{i=1}^{t}(1+\omega(y_{s,i}^{(n)}))$, which satisfies $V_{s,t}(n) = V_{s,t-1}(n)\big(1+\omega(y_{s,t}^{(n)})\big)$ for $t>1$ and $V_{s,1}(n) = \big(1+\omega(y_{s,1}^{(n)})\big)$. We also note that $V_{s,t}(0) = (1+\omega(0))^t = \left(\frac{b^\alpha-1}{b^\alpha-b}\right)^t$, since $y_{s,t}^{(n)} = 0$, independent of the generating vector. This leads to the following decomposition of the error for product weights

$$\begin{aligned} E_{s,t}(\boldsymbol{q}) = E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha}) &+ \frac{\gamma_s}{b^m}\left[(1+\omega(0))^t - 1\right] Y_{s-1}(0) \\ &+ \frac{\gamma_s}{b^m}\sum_{n=1}^{b^m-1}\left(V_{s,t-1}(n)-1\right) Y_{s-1}(n) \\ &+ \frac{\gamma_s}{b^m}\sum_{n=1}^{b^m-1}\omega(y_{s,t}^{(n)}) V_{s,t-1}(n) Y_{s-1}(n), \end{aligned} \tag{12}$$

where only (12) depends on the unknown $q_{s,t}$. This reformulation allows efficient computation of $E_{s,t}$ during the CBC construction by updating intermediate quantities.

## 3.2 SPOD Weights

The search criterion (9) can be reformulated to obtain [4, 3.43]

$$E_{s,t}(\boldsymbol{q}) = \frac{1}{b^m}\sum_{n=0}^{b^m-1}\sum_{\ell=1}^{\alpha s}\ell! \sum_{\substack{\nu\in\{0:\alpha\}^s\\ |\nu|=\ell}}\Big(\prod_{\substack{j=1\\ \nu_j>0}}^{s}\gamma_j(\nu_j)\Big)\sum_{\substack{\mathfrak{v}\subseteq\{1:d\}\text{ s.t.}\\ \mathfrak{u}(\mathfrak{v})=\{1\le j\le s:\nu_j>0\}}}\prod_{j\in\mathfrak{v}}\omega(y_j^{(n)}). \tag{13}$$

For a complete block (i.e. $t=\alpha$), we write $E_{s,\alpha}(\boldsymbol{q}) = \frac{1}{b^m}\sum_{n=0}^{b^m-1}\sum_{\ell=1}^{\alpha s} U_{s,\ell}(n)$, where $U_{s,\ell}(n)$ is given by

$$U_{s,\ell}(n) = \ell! \sum_{\substack{\nu\in\{0:\alpha\}^s\\ |\nu|=\ell}}\prod_{\substack{j=1\\ \nu_j>0}}^{s}\Big[\gamma_j(\nu_j)\Big(\prod_{i=1}^{\alpha}\big(1+\omega(y_{j,i}^{(n)})\big)-1\Big)\Big].$$

Proceeding as in the product weight case, we separate out a $E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha})$ term by

$$E_{s,t}(\boldsymbol{q}) = E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha}) + \frac{1}{b^m}\sum_{n=0}^{b^m-1}\Big(\prod_{i=1}^{t}(1+\omega(y_{s,i}^{(n)}))-1\Big)\Big(\sum_{\ell=1}^{\alpha s}\sum_{\nu_s=1}^{\min(\alpha,\ell)}\gamma_s(\nu_s)\frac{\ell!}{(\ell-\nu_s)!}U_{s-1,\ell-\nu_s}(n)\Big).$$

Defining $V_{s,t}(n)$ as above and with $W_s(n) = \sum_{\ell=1}^{\alpha s}\sum_{\nu_s=1}^{\min(\alpha,\ell)}\gamma_s(\nu_s)\frac{\ell!}{(\ell-\nu_s)!}U_{s-1,\ell-\nu_s}(n)$, we again aim to isolate the term depending on the unknown $q_{s,t}$. This yields

$$E_{s,t}(\boldsymbol{q}) = E_{s-1,\alpha}(\boldsymbol{q}_{s-1,\alpha}) + \frac{1}{b^m}\Big(\Big(\frac{b^\alpha-1}{b^\alpha-b}\Big)^t-1\Big)W_s(0) + \frac{1}{b^m}\sum_{n=1}^{b^m-1}(V_{s,t-1}(n)-1)W_s(n) \tag{14}$$

$$+ \frac{1}{b^m}\sum_{n=1}^{b^m-1}V_{s,t-1}(n)W_s(n)\omega(y_{s,t}^{(n)}), \tag{15}$$

where only the last sum (15) depends on $q_{s,t}$ through $y_{s,t}^{(n)}$.

As an optimization, the remaining terms can be ignored, since the error $E(\boldsymbol{q}_{d-1},z)$ is shifted by the same amount for all candidates $z \in G_{b,m}$. This optimization saves $\mathcal{O}(N)$ operations due to the omission of the sum (14). An analogous optimization is possible in the product weight case. Since the value of the error bound is sometimes a useful quantity, implementations may choose to compute the full bounds from above.

### *3.3 Efficient Implementation*

As currently written, the evaluation of the sums (12) and (15) for all possible $b^m-1$ values for $q_{s,t}$ requires $\mathcal{O}(N^2)$ operations. We view this sum as a matrix-vector multiplication of the matrix

$$\boldsymbol{\Omega} := \left[\omega\left(v_m\left(\frac{n(x)q(x)}{P(x)}\right)\right)\right]_{\substack{1\le n\le b^m-1\\ q\in G_{b,m}}} \tag{16}$$

with the vector consisting of the component-wise product $\big[V_{s,t-1}(n)W_s(n)\big]_{1\le n\le b^m-1}$.

The elements of $\boldsymbol{\Omega}$ depend on $n(x)q(x)$, which is a product of polynomials in $G_{b,m}$. Since the nonzero elements of a finite field form a cyclic group under multiplication, there exists a primitive element $g$ that generates the group, i.e. every element of $G_{b,m}$ can be given as some exponent of $g$.

By using the so-called Rader transform, originally developed in [12], the rows and columns of $\boldsymbol{\Omega}$ can be permuted to obtain a *cyclic* matrix $\boldsymbol{\Omega}^{\text{perm}}$. Application of the fast Fourier transform allows the multiplications (12) and (15) to be executed in $\mathcal{O}(N\log N)$ operations. This technique was applied to Component-by-Component construction techniques by [11]; we also mention the exposition in [5, Ch. 10.3].

### *3.4 Algorithms*

In Algorithms 1 and 2 below, $\mathsf{V}, \mathsf{W}, \mathsf{Y}, \mathsf{U}(\ell)$ and $\mathsf{X}(\ell)$ denote vectors of length $N$. $\mathsf{E}$ is a vector of length $N-1$ and $E_{\text{old}}, E_1, E_2$ are scalars. By $\odot$ we denote component-wise multiplication and $\boldsymbol{\Omega}_{z,:}$ denotes the $z$-th row of $\boldsymbol{\Omega}$.

**Algorithm 1** CBC_product$(b, m, \alpha, s_{max}, \{\gamma_1, \ldots, \gamma_s\})$

$\mathsf{Y} \leftarrow \mathbf{1}$
$E_{\text{old}} \leftarrow 0$
**for** $s = 1, \ldots, \alpha s_{max}$ **do**
  $\mathsf{V} \leftarrow \mathbf{1}$
  **for** $t = 1, \ldots, \alpha$ **do**
    $E_1 \leftarrow \frac{\gamma_s}{b^m}\left(\left(\frac{b^\alpha - 1}{b^\alpha - b}\right)^t - 1\right)\mathsf{Y}(0)$
    $E_2 \leftarrow \frac{\gamma_s}{b^m}\sum_{n=1}^{b^m-1}\left(\mathsf{V}(n) - 1\right)\mathsf{Y}(n)$
    $\mathsf{E} \leftarrow \boldsymbol{\Omega} \cdot (\mathsf{V} \odot \mathsf{Y}) + (E_{\text{old}} + E_1 + E_2) \cdot \mathbf{1}$
    $q_{s,t} \leftarrow \operatorname{argmin}_{q \in G_{b,m}} \mathsf{E}(q)$
    $\mathsf{Y} \leftarrow \left(1 + \gamma_s(\mathsf{V} - 1)\right) \odot \mathsf{Y}$
  **end for**
  $E_{\text{old}} \leftarrow \mathsf{E}(q_{s,\alpha})$
**end for**
**return** $\boldsymbol{q}, E_{\text{old}}$

**Algorithm 2** CBC_SPOD$(b, m, \alpha, s_{max}, \{\gamma_j(\cdot)\}_{j=1}^{s})$

$\mathsf{U}(0) \leftarrow \mathbf{1}$, $\mathsf{U}(1 : \alpha s_{max}) \leftarrow \mathbf{0}$
$E_{\text{old}} \leftarrow 0$
**for** $s = 1, \ldots, \alpha s_{max}$ **do**
  $\mathsf{V} \leftarrow \mathbf{1}$
  $\mathsf{W} \leftarrow \mathbf{0}$
  **for** $\ell = 1, \ldots, s$ **do**
    $\mathsf{X}(\ell) \leftarrow 0$
    **for** $\nu = 1, \ldots, \min(\alpha, \ell)$ **do**
      $\mathsf{X}(\ell) \leftarrow \mathsf{X}(\ell) + \gamma_s(\nu)\frac{\ell!}{(\ell-\nu)!}\mathsf{U}(\ell - \nu)$
    **end for**
    $\mathsf{W} \leftarrow \mathsf{W} + \frac{1}{b^m}\mathsf{X}(\ell)$
  **end for**
  **for** $t = 1, \ldots, \alpha$ **do**
    $E_1 \leftarrow \left(\left(\frac{b^\alpha - 1}{b^\alpha - b}\right)^t - 1\right)\mathsf{W}(0)$
    $E_2 \leftarrow \sum_{n=1}^{b^m-1}\left(\mathsf{V}(n) - 1\right)\mathsf{W}(n)$
    $\mathsf{E} \leftarrow \boldsymbol{\Omega} \cdot (\mathsf{V} \odot \mathsf{W}) + (E_{\text{old}} + E_1 + E_2) \cdot \mathbf{1}$
    $q_{s,t} \leftarrow \operatorname{argmin}_{q \in G_{b,m}} \mathsf{E}(q)$
    $\mathsf{V} \leftarrow (1 + \boldsymbol{\Omega}_{q_{s,t},:}) \odot \mathsf{V}$
  **end for**
  $E_{\text{old}} \leftarrow \mathsf{E}(q_{s,\alpha})$
  **for** $\ell = 1, \ldots, \alpha s$ **do**
    $\mathsf{U}(\ell) \leftarrow \mathsf{U}(\ell) + (\mathsf{V} - 1) \odot \mathsf{X}(\ell)$
  **end for**
**end for**
**return** $\boldsymbol{q}, E_{\text{old}}$

# 4 Implementation Considerations

## 4.1 Walsh Coefficient Bound

The definition of the auxiliary weights (6) and (8) contain powers of the constant $C_{\alpha,b}$ defined in (4), which for $b = 2$ is bounded from below by $C_{\alpha,2} = \frac{9}{2}\left(\frac{5}{3}\right)^{\alpha-2} \geq \frac{9}{2}$. The resulting large values of the worst-case error bounds (7) and (9) have been found to lead to generating vectors with bad projections. For integrand functions with small Walsh coefficients, $C_{\alpha,b}$ may be replaced with a tighter bound $C$; this will yield a worst-case error bound better adapted to the integrand and a generating vector with the desired properties. Since additionally $C_{\alpha,b}$ is increasing in $\alpha$ for fixed $b$, this becomes more important as the order of the quadrature rule increases.

## 4.2 Pruning

For large values of the WCE, the elements of the generating vector can repeat, leading to very bad projections in certain dimensions. For standard lattice rules, if $q_s = q_{\tilde{s}}$ for two dimensions $s$ and $\tilde{s}$, the corresponding components of the quadrature points will be identical, $x_s^{(n)} = x_{\tilde{s}}^{(n)}$ for all values of $n = 0, \ldots, b^m - 1$. Thus, in the projection onto the $(s, \tilde{s})$-plane, only points on the diagonal are obtained. This is obviously a very bad choice. For polynomial lattice rules, a similar effect can be observed.

With the aim of alleviating this effect, we formulate a pruning procedure that incorporates this observation into the construction of the generating vector. We impose the additional condition that the newest element of the generating vector is unique, i.e. is not equal to a previously constructed component of $\boldsymbol{q}$. This can be achieved in the CBC construction by replacing the minimization of $E(\boldsymbol{q})$ over all possible $b^m - 1$ values of the new component by the restricted version

$$q_d = \operatorname*{argmin}_{\substack{z \in G_{b,m}, \\ z \notin \{q_1, \ldots, q_{d-1}\}}} E(q_1, \ldots, q_{d-1}, z).$$

This procedure requires $d - 1$ operations in iteration $d$ to check the previous entries of the vector, or $\mathcal{O}(\alpha^2 s^2)$ in total, and thus does not increase the asymptotic complexity. Alternatively, the indices can be stored in an additional sorted data structure with logarithmic (in $\alpha s$) cost for both inserting new indices and checking for membership. This yields a cost of $\mathcal{O}(\alpha s \log(\alpha s))$ additional operations, with an additional storage cost of $\mathcal{O}(\alpha s)$.

# 5 Results

## 5.1 Model Problems

For our numerical results, we consider two problems, one of SPOD-type weights and one with product weights. The SPOD-type integrand we consider was first mentioned in [8], and models a parametric partial differential equation depending in an affine manner on $s$ parameters $y_1, \ldots, y_s$:

$$f_{\theta,s,\zeta}(\boldsymbol{y}) = \left(1 + \theta \cdot \sum_{j=1}^{s} a_j y_j\right)^{-1}, \quad a_j = j^{-\zeta}. \tag{17}$$

We have the differential $\partial_{\boldsymbol{y}}^{\boldsymbol{\nu}} f_{\theta,s,\zeta}(\boldsymbol{y}) = (-1)^{|\boldsymbol{\nu}|}|\boldsymbol{\nu}|! f_{\theta,s,\zeta}^{|\boldsymbol{\nu}|+1}(\boldsymbol{y}) \prod_{j=1}^{s} (\theta a_j)^{\nu_j}$, leading to the bound

$$\forall \boldsymbol{\nu} \in \{0,1,\ldots,\alpha\}^s : \quad |\partial_{\boldsymbol{y}}^{\boldsymbol{\nu}} f_{\theta,s,\zeta}(\boldsymbol{y})| \le C_f |\boldsymbol{\nu}|! \prod_{j=1}^{s} \beta_j^{\nu_j},$$

for a $C_f \ge 1$ and with the weights $\beta_j$ given by

$$\beta_j = \theta a_j = \theta j^{-\zeta}, \quad j = 1, \ldots, s. \tag{18}$$

Additionally, for $s \to \infty$, we have $(\beta_j)_j \in \ell^p(\mathbb{N})$ with $p > \frac{1}{\zeta}$ and thus $\alpha = \lfloor 1/p \rfloor + 1 = \zeta$. Therefore, by Theorem 3.2 of [4], we have that an interlaced polynomial lattice rule of order $\alpha$ with $N = b^m$ points ($b$ prime, $m \ge 1$) and point set $\mathscr{P}_N$ can be constructed such that the QMC quadrature error fulfills

$$|\mathscr{I}[f_{\theta,s,\zeta}] - \mathscr{Q}_{\mathscr{P}_N}[f_{\theta,s,\zeta}]| \le C(\alpha, \boldsymbol{\beta}, b, p) N^{-1/p},$$

for a constant $C(\alpha, \boldsymbol{\beta}, b, p)$ independent of $s$ and $N$. We also consider separable integrand functions, which, on account of their separability, trivially belong to the product weight class. They are given by

$$g_{\theta,s,\zeta}(\boldsymbol{y}) = \exp\left(\theta \sum_{j=1}^{s} a_j y_j\right), \quad a_j = j^{-\zeta}, \tag{19}$$

and satisfy $\partial_{\boldsymbol{y}}^{\boldsymbol{\nu}} g(\boldsymbol{y}) = g(\boldsymbol{y}) \prod_{k=1}^{s} (\theta a_k)^{\nu_k}$. Under the assumption that there exists a constant $\widetilde{C} > 0$ that is independent of $s$ and such that $g(\boldsymbol{y}) \le \widetilde{C}$ for all $\boldsymbol{y} \in [0,1]^s$, which holds here with $\widetilde{C} = \exp(\theta \sum_{j=1}^{s} j^{-\zeta})$, $\zeta > 1$, we have the bound

$$\forall \boldsymbol{\nu} \in \{0,1,\ldots,\alpha\}^s : \quad |\partial_{\boldsymbol{y}}^{\boldsymbol{\nu}} g_{\theta,s,\zeta}(\boldsymbol{y})| \le C_g \prod_{j=1}^{s} \beta_j^{\nu_j},$$

for a $C_g > \widetilde{C}$ and with the weights $\beta_j$ given by $\beta_j = \theta a_j = \theta j^{-\zeta}$ for $j = 1, \ldots, s$, as in (18). We have the following analytically given formula for the integral

$$\mathscr{I}[g_{\theta,s,\zeta}] = \prod_{j=1}^{s} \left[ \frac{j^{\zeta}}{\theta} \left( \exp(\theta j^{-\zeta}) - 1 \right) \right] = \exp \left[ \sum_{j=1}^{s} \log \left( \sum_{\mu=0}^{\infty} \frac{\left(\theta j^{-\zeta}\right)^{\mu}}{(\mu+1)!} \right) \right], \quad (20)$$

along with the quadrature error bound for an interlaced polynomial point set $\mathscr{P}_N$

$$|\mathscr{I}[g_{\theta,s,\zeta}] - \mathscr{Q}_{\mathscr{P}_N}[g_{\theta,s,\zeta}]| \leq C(\alpha, \boldsymbol{\beta}, b, p) N^{-1/p}.$$

## 5.2 Validation of Work Bound

The timing results in Figure 1 show that the work bounds $\mathscr{O}(\alpha s N \log N + \alpha^2 s^2 N)$ for SPOD weights from [4, Thm. 3.1] and $\mathscr{O}(\alpha s N \log N)$ for product weights from [4, Thm. 3.2] are fulfilled in practice and seem to be tight. The work $\mathscr{O}(N \log N)$ in the number of QMC points $N$ also appears tight for moderate $s$ and $N$.

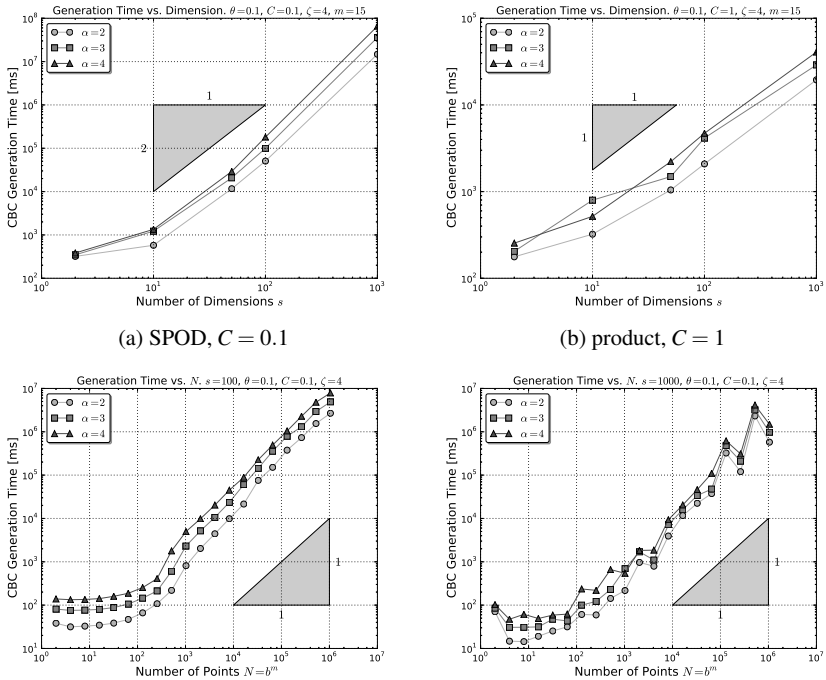
## 5.3 Pruning and Adapting the Walsh Coefficient Bound

We consider construction of the generating vector with and without application of the pruning procedure defined in Section 4.2. Convergence rates for both cases can be seen in Figure 2: for $\alpha = 2$ no difference was observed when pruning the entries.

Results for the constant $C_{\alpha,b}$ from (4) as well as for $C = 1$ are shown; in this example, adapting the constant $C$ to the integrand seems to yield better results than pruning. In the case of the integrand (17), this can be justified by estimating the Walsh coefficients by numerical computation of the Walsh-Hadamard transform. The maximal values of these numerically computed coefficients is bounded by 1 for low dimensions, indicating that the bound $C_{\alpha,b}$ is too pessimistic.

## 5.4 Higher-Order Convergence

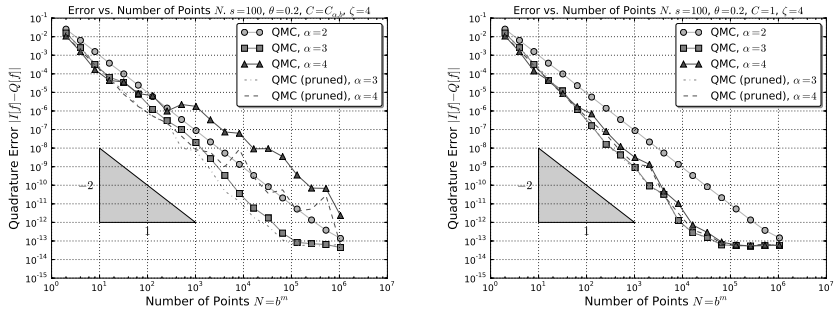
As can be seen in figures 3 and 4, the higher-order convergence rates proved in [4] can be observed in practice for the two tested integrands. Here, $C = 0.1$ was used as the Walsh coefficient bound. We also mention that for more general, non-affine, holomorphic parameter dependence of operators the same convergence rates and derivative bounds as in [4] have been recently established in [3]. The present CBC constructions apply also to these (non affine-parametric) problems.




(a) SPOD, $C = 0.1$ (b) product, $C = 1$

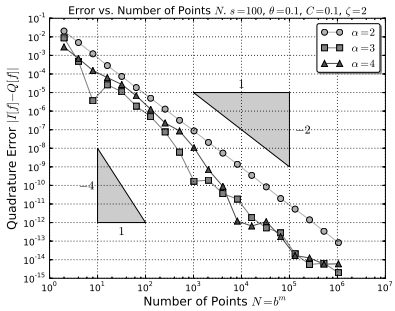
(c) SPOD, $s = 100$ (d) product, $s = 1000$

Fig. 1: CPU time required for the construction of generating vectors of varying order $\alpha = 2,3,4$ for product and SPOD weights vs. the dimension $s$ in (a) and (b) and vs. the number of points $N = 2^m$ in (c) and (d).

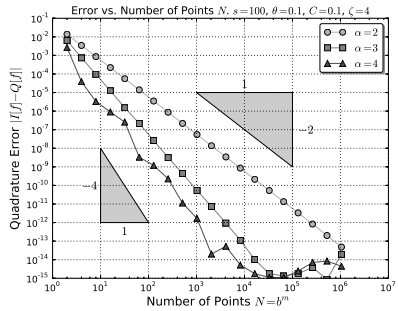



(a) $C = C_{\alpha,b}$ (b) $C = 1$

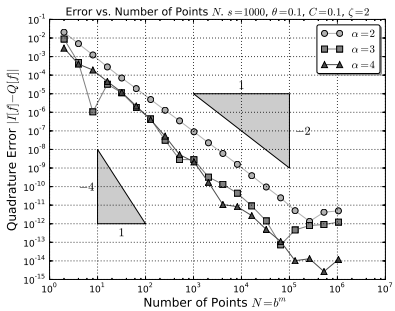
Fig. 2: Convergence of QMC approximation for the SPOD integrand (17) in $s = 100$ dimensions with interlacing parameter $\alpha = 2,3,4$, with and without pruning.



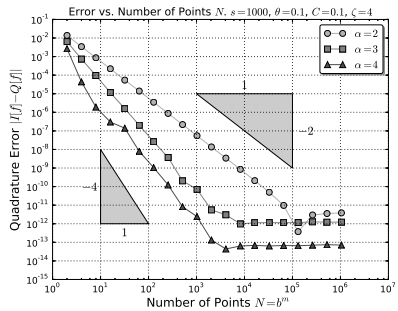

(a) $s = 100$, $\zeta = 2$




(b) $s = 100$, $\zeta = 4$

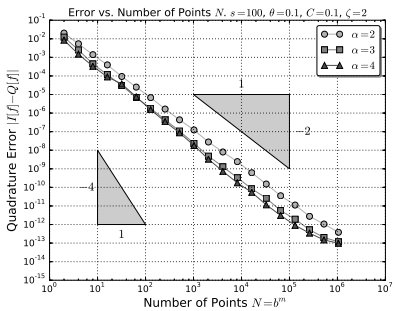



(c) $a = 1000$, $\zeta = 2$

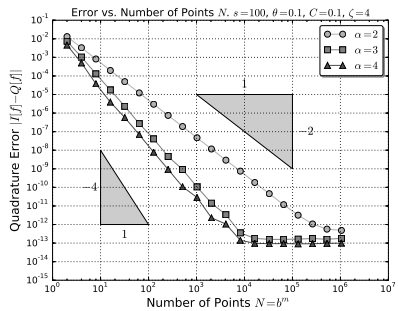



(d) $s = 1000$, $\zeta = 4$

Fig. 3: Convergence of QMC approximation to (20) for the product weight integrand (19) in $s = 100, 1000$ dimensions with interlacing parameter $\alpha = 2, 3, 4$.




(a) $\zeta = 2$




(b) $\zeta = 4$

Fig. 4: Convergence of QMC approximation for the SPOD weight integrand (17) in $s = 100$ dimensions with interlacing parameter $\alpha = 2, 3, 4$.

**Acknowledgements** This work is supported by the Swiss National Science Foundation (SNF) under project number 200021_149819 and by the European Research Council (ERC) under FP7 Grant AdG247277. Work of CS performed in part while CS visited ICERM / Brown University in September 2014; the excellent ICERM working environment is warmly acknowledged.

## References

1. Cools, R., Kuo, F.Y., Nuyens, D.: Constructing embedded lattice rules for multivariable integration. SIAM J. Sci. Comput. **28**(6), 2162–2188 (electronic) (2006). DOI 10.1137/06065074X
2. Dick, J.: The decay of the Walsh coefficients of smooth functions. Bull. Aust. Math. Soc. **80**(3), 430–453 (2009). DOI 10.1017/S0004972709000392
3. Dick, J., Gia, Q.T.L., Schwab, C.: Higher-order quasi-Monte Carlo integration for holomorphic, parametric operator equations. Tech. Rep. 2014-23, Seminar for Applied Mathematics, ETH Zürich (2014)
4. Dick, J., Kuo, F., Gia, Q.T.L., Nuyens, D., Schwab, C.: Higher order QMC Galerkin discretization for parametric operator equations. SIAM J. Numerical Analysis (submitted 2013)
5. Dick, J., Pillichshammer, F.: Digital nets and sequences. Cambridge University Press, Cambridge (2010). DOI 10.1017/CBO9780511761188
6. Goda, T., Dick, J.: Construction of interlaced scrambled polynomial lattice rules of arbitrary high order. arXiv preprint arXiv:1301.6441 (2013)
7. Kuo, F.Y.: Component-by-component constructions achieve the optimal rate of convergence for multivariate integration in weighted Korobov and Sobolev spaces. J. Complexity **19**(3), 301–320 (2003). DOI 10.1016/S0885-064X(03)00006-2
8. Kuo, F.Y., Schwab, C., Sloan, I.H.: Quasi-Monte Carlo methods for high-dimensional integration: the standard (weighted Hilbert space) setting and beyond. The ANZIAM Journal **53**, 1–37 (2011). DOI 10.1017/S1446181112000077
9. Niederreiter, H.: Random number generation and quasi-Monte Carlo methods, *CBMS-NSF Regional Conference Series in Applied Mathematics*, vol. 63. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (1992). DOI 10.1137/1.9781611970081
10. Nuyens, D., Cools, R.: Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. Math. Comp. **75**(254), 903–920 (electronic) (2006). DOI 10.1090/S0025-5718-06-01785-6
11. Nuyens, D., Cools, R.: Fast component-by-component construction, a reprise for different kernels. In: Monte Carlo and quasi-Monte Carlo methods 2004, pp. 373–387. Springer, Berlin (2006). DOI 10.1007/3-540-31186-6_22
12. Rader, C.: Discrete Fourier transforms when the number of data samples is prime. Proceedings of the IEEE **3**(3), 1–2 (1968)
13. Sloan, I.H., Kuo, F.Y., Joe, S.: Constructing randomly shifted lattice rules in weighted Sobolev spaces. SIAM J. Numer. Anal. **40**(5), 1650–1665 (2002). DOI 10.1137/S0036142901393942
14. Sloan, I.H., Reztsov, A.V.: Component-by-component construction of good lattice rules. Math. Comp. **71**(237), 263–273 (2002). DOI 10.1090/S0025-5718-01-01342-4

# Recent Research Reports

| Nr. | Authors/Title |
|---|---|
| 2014-15 | Ch. Schwab<br>Exponential convergence of simplicial hp-FEM for H ¹-functions with isotropic singularities |
| 2014-16 | P. Grohs and S. Keiper and G. Kutyniok and M. Schaefer<br>$\alpha$-Molecules |
| 2014-17 | A. Hiltebrand and S. Mishra<br>Efficient computation of all speed flows using an entropy stable shock-capturing space-time discontinuous Galerkin method |
| 2014-18 | D. Conus and A. Jentzen and R. Kurniawan<br>Weak convergence rates of spectral Galerkin approximations for SPDEs with nonlinear diffusion coefficients |
| 2014-19 | J. Doelz and H. Harbrecht and Ch. Schwab<br>Covariance regularity and H-matrix approximation for rough random fields |
| 2014-20 | P. Grohs and S. Hosseini<br>Nonsmooth Trust Region Algorithms for Locally Lipschitz Functions on Riemannian Manifolds |
| 2014-21 | P. Grohs and A. Obermeier<br>Optimal Adaptive Ridgelet Schemes for Linear Transport Equations |
| 2014-22 | S. Mishra and Ch. Schwab and J. Sukys<br>Multi-Level Monte Carlo Finite Volume methods for uncertainty quantification of acoustic wave propagation in random heterogeneous layered medium |
| 2014-23 | J. Dick and Q. T. Le Gia and Ch. Schwab<br>Higher order Quasi Monte Carlo integration for holomorphic, parametric operator equations |
| 2014-24 | C. Sanchez-Linares and M. de la Asunci on and M. Castro and S. Mishra and J. Šukys<br>Multi-level Monte Carlo finite volume method for shallow water equations with uncertain parameters applied to landslides-generated tsunamis |