

Computational Higher Order Quasi-Monte Carlo Integration

R. N. Gantner and Ch. Schwab

Research Report No. 2014-25

September 2014

Latest revision: September 2015

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

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 [6] is considered and the computational performance of these higher-order QMC rules is investigated on a suite of parametric, high-dimensional test integrand functions. After reviewing the principles of their construction by the “fast component-by-component” (CBC) algorithm due to Nuyens and Cools as well as recent theoretical results on their convergence rates from [Dick, J., Kuo, F. Y., Le Gia, Q. T., Nuyens, D., Schwab, C.: Higher order QMC Petrov-Galerkin discretization for affine parametric operator equations with random field inputs. *SIAM J. Numer. Anal.* **52**(6) (2014), pp. 2676-2702], we indicate algorithmic aspects and implementation details of their efficient construction. Instances of higher order QMC quadrature rules are applied to several 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) estimates on the Walsh coefficients of the integrand provide quantitative improvements of the convergence behavior. The sharpness of theoretical, asymptotic bounds on memory usage and operation counts, with respect to the number of QMC points N and to the dimension s of the integration domain is verified experimentally to hold starting with dimension as low as $s = 10$ and with $N = 128$. The efficiency of the proposed algorithms for computation of the generating vectors is investigated for the considered classes of functions in dimensions $s = 10, \dots, 1000$. A pruning procedure for components of the generating vector is proposed and computationally investigated. The use of pruning is shown to yield quantitative improvements in the QMC error, but also to not affect the asymptotic convergence rate, consistent with recent theoretical findings from [Dick, J., Kritzer, P.: On a projection-corrected component-by-component construction. *Journal of Complexity* (2015) DOI 10.1016/j.jco.2015.08.001].

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

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. In particular, 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 implicitly lower-dimensional structure in parametric input data and solutions of such PDEs. 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 provided that the variances are bounded independent of the dimension, but it is limited to $1/2$. Thus it is important to devise integration methods which converge of higher order than $1/2$, independent of the dimension of the integration domain.

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 [6, 11, 10], 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 $f : [0, 1]^s \rightarrow \mathbb{R}$ of s variables y_1, \dots, y_s over the s -dimensional unit cube,

$$\mathcal{I}[f] := \int_{[0,1]^s} f(y_1, \dots, y_s) dy_1 \cdots dy_s. \quad (1)$$

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 smoothness-driven product-and-order-dependent (SPOD) type. Denoting the interlaced polynomial lattice point set by $\mathcal{P} = \{\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(N-1)}\}$ with $\mathbf{x}^{(n)} \in [0, 1]^s$ for $n = 0, \dots, N-1$, we write the QMC quadrature rule as

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

In Section 2 we first define in more detail the structure of the point set \mathcal{P} considered throughout and derive worst-case error bounds for integrand functions which belong to certain weighted spaces of functions introduced in [13]. Then, the component-by-component construction is 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 imple-

mentation and application of these rules. In Section 5, we give measured convergence results for several model integrands, showing the applicability of these methods.

2 Interlaced Polynomial Rank-1 Lattice Rules

Polynomial rank-1 lattice point sets, introduced by Niederreiter in [14], are a modification of standard rank-1 lattice point sets to polynomial arithmetic in $\mathbb{Z}_b[x]$ (defined in the next section). 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 [6, Def. 3.6], [7, Def. 5.1] and focus on computational techniques for their efficient construction.

2.1 Definitions

For a given prime number b , let \mathbb{Z}_b denote the finite field of order b and $\mathbb{Z}_b[x]$ the set of polynomials with coefficients in \mathbb{Z}_b . Let $P \in \mathbb{Z}_b[x]$ be an irreducible polynomial of degree m . Then, the finite field of order b^m is isomorphic to the residue class $(\mathbb{Z}_b[x]/P, +, \cdot)$, where both operations are carried out in $\mathbb{Z}_b[x]$ modulo P . We denote by $G_{b,m} = ((\mathbb{Z}_b[x]/P)^*, \cdot)$ the cyclic group formed by the nonzero elements of the residue class together with polynomial multiplication modulo P .

Throughout, we frequently interchange an integer n , $0 \leq n < N = b^m$, with its associated polynomial $n(x) = \eta_0 + \eta_1 x + \eta_2 x^2 + \dots + \eta_{m-1} x^{m-1}$, the coefficients of which are given by the b -adic expansion $n = \eta_0 + \eta_1 b + \eta_2 b^2 + \dots + \eta_{m-1} b^{m-1}$.

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 \mathcal{P} :

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

where the mapping $v_m : \mathbb{Z}_b((x^{-1})) \rightarrow [0, 1)$ is given for any integer w by the expression $v_m \left(\sum_{\ell=w}^{\infty} \xi_{\ell} x^{-\ell} \right) = \sum_{\ell=\min(1,w)}^m \xi_{\ell} b^{-\ell}$, and $\mathbb{Z}_b((x^{-1}))$ denotes the set of formal Laurent series $\sum_{k=w}^{\infty} a_k x^{-k}$ with $a_k \in \mathbb{Z}_b$ for some integer w .

A key ingredient for obtaining QMC formulas which afford higher-order convergence rates is the *interlacing* of lattice point sets, as introduced in [1, 2]. We define the digit interlacing function, which maps α points in $[0, 1)$ to one point in $[0, 1)$.

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

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

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

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

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

i.e. the i -th coordinate of the n -th point is obtained by interlacing a block of α coordinates.

2.2 Worst-Case Error Bound

We give here an overview of bounds on the worst case error which are required for the CBC construction; for details we refer to [6]. The results therein were based on a “new function space setting”, which generalizes the notion of a *reproducing kernel Hilbert space* to a Banach space setting. We also refer to [13] for an overview of related function spaces.

2.2.1 Function Space Setting

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

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

with the obvious modifications if q or r is infinite. Here $\{1:s\}$ is a shorthand notation for the set $\{1, 2, \dots, s\}$, and $(\boldsymbol{\alpha}_{\mathbf{v}}, \boldsymbol{\tau}_{\mathbf{u} \setminus \mathbf{v}}, \mathbf{0})$ denotes a sequence \mathbf{v} with $v_j = \alpha$ for $j \in \mathbf{v}$, $v_j = \tau_j$ for $j \in \mathbf{u} \setminus \mathbf{v}$, and $v_j = 0$ for $j \notin \mathbf{u}$. For non-negative weights $\gamma_{\mathbf{u}}$, the space $\mathscr{W}_{s,\alpha,\gamma,q,r}$ consists of smooth functions with integrable mixed derivatives of orders up to α with respect to each variable, and L^q -integrable ($q \in [1, \infty]$) mixed derivatives containing a derivative of order α in at least one variable.

This space is called *unanchored* because the innermost integral over $[0,1]^{s-|\mathbf{v}|}$ in the definition of the norm $\|\circ\|_{s,\alpha,\gamma,q,r}$ integrates out the “inactive” coordinates, i.e. those with respect to which a derivative of order less than α is taken, rather than “anchoring” these variables by fixing their values equal to an anchor point

$a \in [0, 1)^s$. The weights γ_u in the definition of the norm can be interpreted as the relative importance of groups of variables u . Below, we will assume either product structure or so-called SPOD structure on the weights γ_u ; here, the acronym ‘‘SPOD’’ stands for smoothness-driven, product and order dependent weights, which were first introduced in [6].

We remark that the sum over all subsets $u \subseteq \{1 : s\}$ also includes the empty set $u = \emptyset$, for which we obtain the term $\gamma_\emptyset^{-r} \left| \int_{[0,1]^s} f(\mathbf{y}) d\mathbf{y} \right|^r$, which contains the average of the function f over the s -dimensional unit cube.

2.2.2 Error Bound

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

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

Assume that $1 \leq r, r' \leq \infty$ with $1/r + 1/r' = 1$ and $\alpha, s \in \mathbb{N}$ with $\alpha > 1$. Define a collection of positive weights $\gamma = (\gamma_u)_{u \subset \mathbb{N}}$. Then, by [6, Thm. 3.5], we have the following bound on the worst-case error in the space $\mathcal{W}_{s, \alpha, \gamma, q, r}$,

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

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

$$e_{s, \alpha, \gamma, r'}(\mathcal{P}) = \left(\sum_{\emptyset \neq u \subseteq \{1:s\}} \left(C_{\alpha, b}^{|u|} \gamma_u \sum_{k_u \in \mathcal{D}_u^*} b^{-\mu_\alpha(k_u)} \right)^{r'} \right)^{1/r'}. \quad (3)$$

The inner sum is over all elements of the dual net without zero, see [10, Def. 5]. For a number k with b -adic expansion $k = \sum_{j=1}^J \kappa_j b^{a_j}$ with $a_1 > \dots > a_J$, we define the weight $\mu_\alpha(k) = \sum_{j=1}^{\min(\alpha, J)} (a_j + 1)$ as in [6]. The constant $C_{\alpha, b}$ is obtained by bounding the Walsh coefficients of functions in Sobolev spaces, see [3, Thm. 14] for details. Here, it has the value

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

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 ℓ^∞ norm over the sequence indexed by $u \subseteq \{1 : s\}$ in the norm $\|\circ\|_{s, \alpha, \gamma, q, r}$). We

denote by $\widetilde{\mathcal{P}}$ a point set in αs dimensions, and use in the following the definition $\omega(y) = \frac{b-1}{b^\alpha - b} - b^{\lfloor \log_b y \rfloor (\alpha-1)} \frac{b^\alpha - 1}{b^\alpha - b}$ where $\omega(0) = \frac{b-1}{b^\alpha - b}$. Using [6, Thm. 3.9], we bound the sum over the dual net \mathcal{D}_u^* in (3) by a computationally amenable expression,

$$e_{s,\alpha,\gamma,1}(\mathcal{P}) \leq E_{\alpha s}(\mathbf{q}) = \frac{1}{b^m} \sum_{n=0}^{b^m-1} \sum_{\substack{\mathbf{v} \subseteq \{1:\alpha s\} \\ \mathbf{v} \neq \emptyset}} \widetilde{\gamma}_{\mathbf{v}} \prod_{j \in \mathbf{v}} \omega(y_j^{(n)}), \quad \mathbf{y}^{(n)} \in \widetilde{\mathcal{P}}, \quad (5)$$

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 $\widetilde{\gamma}_{\mathbf{v}}$, $\mathbf{v} \subseteq \{1:\alpha s\}$ depends on the choice of weights $\gamma_{\mathbf{u}}$.

Assume given a sequence $(\beta_j)_j \in \ell^p(\mathbb{N})$ for $0 < p < 1$ and denote by $\mathbf{u}(\mathbf{v}) \subseteq \{1:s\}$ an ‘‘indicator set’’ containing a dimension $i \in \{1,\dots,s\}$ if any of the corresponding α dimensions $\{(i-1)\alpha+1,\dots,i\alpha\}$ is in $\mathbf{v} \subseteq \{1:\alpha s\}$. This can be given explicitly by $\mathbf{u}(\mathbf{v}) = \{\lfloor j/\alpha \rfloor : j \in \mathbf{v}\}$. For product weights, we define

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

and obtain from (5) the worst-case error bound for $d = 1, \dots, \alpha s$

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

For SPOD weights we have

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

for which we obtain

$$E_d(\mathbf{q}) = \frac{1}{b^m} \sum_{n=0}^{b^m-1} \sum_{\substack{\mathbf{v} \subseteq \{1:d\} \\ \mathbf{v} \neq \emptyset}} \sum_{\mathbf{v} \in \{1:\alpha\}^{|\mathbf{u}(\mathbf{v})|}} |\mathbf{v}|! \left(\prod_{j \in \mathbf{u}(\mathbf{v})} \gamma_j(\mathbf{v}_j) \right) \left(\prod_{j \in \mathbf{v}} \omega(y_j^{(n)}) \right). \quad (9)$$

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 [3].

3 Component-by-Component Construction

The component-by-component construction (CBC) [12, 18, 19] 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 [15, 16] the CBC construction to exploit the cyclic structure inherent in the point sets for standard lattice rules when the number of points N 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

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

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

3.1 Product Weights

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

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

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 component $q_{s,t}$, we denote the truncated generating vector by $\mathbf{q}_d = (q_1, \dots, q_d)$ or in analogy to (10), $\mathbf{q}_{s,t} = (q_{1,1}, \dots, q_{s,t})$. We now write $E_{s,t}(\mathbf{q}_{s,t}) = E_{s-1,\alpha}(\mathbf{q}_{s-1,\alpha}) + \tilde{E}(q_{s,1}, \dots, q_{s,t})$, such that (11) can be used for $E_{s-1,\alpha}(\mathbf{q}_{s-1,\alpha})$ during the iteration over t . For $t < \alpha$, we have

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

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

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

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 $V_{s,t}(n) = V_{s,t-1}(n) (1 + \omega(y_{s,t}^{(n)}))$ for $t > 1$ and $V_{s,1}(n) = (1 + \omega(y_{s,1}^{(n)}))$. We also note that $V_{s,t}(0) = (1 + \omega(0))^t = \left(\frac{b^\alpha - 1}{b^\alpha - b}\right)^t$, since $y_{s,t}^{(0)} = 0$, independent of the generating vector. This leads to the following decomposition of the error for product weights

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

where only (12) depends on the unknown $q_{s,t}$ through $y_{s,t}^{(n)}$. This reformulation permits efficient computation of the worst-case error bound $E_{s,t}$ during the CBC construction by updating intermediate quantities.

3.2 SPOD Weights

The search criterion (9) can be reformulated to obtain [6, 3.43]

$$E_{s,t}(\mathbf{q}) = \frac{1}{b^m} \sum_{n=0}^{b^m-1} \sum_{\ell=1}^{\alpha s} \ell! \sum_{\substack{\mathbf{v} \in \{0:\alpha\}^s \\ |\mathbf{v}|=\ell}} \left(\prod_{j=1}^s \gamma_j(\mathbf{v}_j) \right) \sum_{\substack{\mathbf{v} \subseteq \{1:d\} \text{ s.t.} \\ \mathbf{u}(\mathbf{v}) = \{1 \leq j \leq s: \mathbf{v}_j > 0\}}} \prod_{j \in \mathbf{v}} \omega(y_j^{(n)}). \quad (13)$$

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

$$U_{s,\ell}(n) = \ell! \sum_{\substack{\mathbf{v} \in \{0:\alpha\}^s \\ |\mathbf{v}|=\ell}} \prod_{j=1}^s \left[\gamma_j(\mathbf{v}_j) \left(\prod_{i=1}^{\alpha} (1 + \omega(y_{j,i}^{(n)})) - 1 \right) \right].$$

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

$$\begin{aligned} E_{s,t}(\mathbf{q}) &= E_{s-1,\alpha}(\mathbf{q}_{s-1,\alpha}) \\ &\quad + \frac{1}{b^m} \sum_{n=0}^{b^m-1} \left(\prod_{i=1}^t (1 + \omega(y_{s,i}^{(n)})) - 1 \right) \left(\sum_{\ell=1}^{\alpha s} \sum_{\mathbf{v}_s=1}^{\min(\alpha,\ell)} \gamma_s(\mathbf{v}_s) \frac{\ell!}{(\ell - \mathbf{v}_s)!} U_{s-1,\ell-\mathbf{v}_s}(n) \right). \end{aligned}$$

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

$$E_{s,t}(\mathbf{q}) = E_{s-1, \alpha}(\mathbf{q}_{s-1, \alpha}) + \frac{1}{b^m} \left(\left(\frac{b^\alpha - 1}{b^\alpha - b} \right)^t - 1 \right) W_s(0) \\ + \frac{1}{b^m} \sum_{n=1}^{b^m-1} (V_{s,t-1}(n) - 1) W_s(n) \quad (14)$$

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

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

The remaining terms can be ignored, since the error $E(\mathbf{q}_{d-1}, z)$ 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 $E_{s_{\max}, \alpha}(\mathbf{q})$ is sometimes a useful quantity, one may choose to compute the full bounds given 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. Following [15], we view this sum as a matrix-vector multiplication of the matrix

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

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

The elements of $\mathbf{\Omega}$ 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 [17], the rows and columns of $\mathbf{\Omega}$ can be permuted to obtain a *circulant* matrix $\mathbf{\Omega}^{\text{perm}}$. 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 the CBC algorithm in [16]; we also mention the exposition in [8, Ch. 10.3].

The total work complexity is $\mathcal{O}(\alpha s N \log N + \alpha^2 s^2 N)$ for SPOD weights and $\mathcal{O}(\alpha s N \log N)$ for product weights [6, Thms. 3.1, 3.2]. In Section 5, we show measurements of the CBC construction time that indicate that the constants in these asymptotic estimates are small, allowing these methods to be applied in practice.

3.4 Algorithms

In Algorithms 1 and 2 below, $\mathbf{V}, \mathbf{W}, \mathbf{Y}, \mathbf{U}(\ell)$ and $\mathbf{X}(\ell)$ denote vectors of length N . \mathbf{E} is a vector of length $N-1$ and E_{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($b, m, \alpha, s_{\text{max}}, \{\gamma_1, \dots, \gamma_s\}$)

```

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

```

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

```

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

```

4 Implementation Considerations

4.1 Walsh Coefficient Bound

The definition of the auxiliary weights (6) and (8) contain powers of the Walsh constant bound $C_{\alpha,b}$ defined in (4), which for $b = 2$ is bounded from below by $C_{\alpha,2} = \frac{9}{2} \left(\frac{5}{3}\right)^{\alpha-2} \geq \frac{9}{2}$. For base $b = 2$, it was recently shown in [20] that $C_{\alpha,2}$ can be replaced by $C = 1$. 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 α 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 polynomial lattice rules, if $q_{s,k} = q_{\tilde{s},k} \forall k = 1, \dots, \alpha$ 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 $n = 0, \dots, b^m - 1$. Thus, in the projection onto the (s, \tilde{s}) -plane, only points on the diagonal are obtained, which is obviously a very bad choice. One way this problem could be avoided is to consider a second error criterion, as in [4]. We propose here a simpler method that requires only minor modification of the CBC iteration.

To alleviate 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 \mathbf{q} . This can be achieved in the CBC construction by replacing the minimization of $E(\mathbf{q})$ over all possible $b^m - 1$ values of the new component by the restricted version

$$q_d = \underset{\substack{z \in G_{b,m}, \\ z \notin \{q_1, \dots, q_{d-1}\}}}{\operatorname{argmin}} E(q_1, \dots, q_{d-1}, z). \quad (17)$$

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 αs) 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 $\mathcal{O}(\alpha s)$. It was shown in [5] that the presently proposed pruning procedure preserves the higher order QMC convergence estimates. In the case where the set of candidates in (17), $G_{b,m} \setminus \{q_1, \dots, q_{d-1}\}$, is empty (which happens e.g. when

$\alpha s_{max} > b^m - 1$), the restriction is dropped. In other words, pruning is applied as long as it still allows for at least one possible value for q_d .

5 Results

We present several tests of an implementation of Algorithms 1 and 2, and of the resulting higher order QMC quadrature rules. Rather than solving concrete application problems, the purpose of the ensuing numerical experiments is a) to verify the validity of the asymptotic (as $s, N \rightarrow \infty$) complexity estimates and QMC error bounds, in particular to determine the range where the asymptotic complexity bounds give realistic descriptions of the CBC construction's performance; b) to investigate the quantitative effect of (not) pruning the generating vector on the accuracy and convergence rates of the QMC quadratures, and c) to verify the necessity of the weighted spaces $\mathcal{W}_{s,\alpha,\gamma,q,r}$ and the norms in (2) for classifying integrand function regularity. We remark that, due to the limited space of these proceedings, only few representative simulations can be presented in detail; for further results and a complete description of our implementation, we refer to [9].

5.1 Model Problems

For our numerical results, we consider two model parametric integrands designed to mimic the behavior of parametric solution families of parametric partial differential equations. Both integrand functions are smooth (in fact, analytic) functions of all integration variables and admit stable extensions to a countable number of integration variables. However, their "sparsity" is controlled, as expressed by the growth of their higher derivatives. The first integrand function belongs to weighted spaces $\mathcal{W}_{s,\alpha,\gamma,q,r}$ with the norms in (2) where the weights are of SPOD type [6], whereas the second integrand allows for product weights. The SPOD-type integrand we consider was first mentioned in [13], and models a parametric partial differential equation depending in an affine manner on s parameters y_1, \dots, y_s , as considered, for example, in [6]:

$$f_{\theta,s,\zeta}(\mathbf{y}) = \left(1 + \theta \cdot \sum_{j=1}^s a_j y_j \right)^{-1}, \quad a_j = j^{-\zeta}, \quad \zeta \in \mathbb{N}. \quad (18)$$

We have the differential $\partial_{\mathbf{y}}^{\mathbf{v}} f_{\theta,s,\zeta}(\mathbf{y}) = (-1)^{|\mathbf{v}|} |\mathbf{v}|! f_{\theta,s,\zeta}^{|\mathbf{v}|+1}(\mathbf{y}) \prod_{j=1}^s (\theta a_j)^{v_j}$, leading to the bound $|\partial_{\mathbf{y}}^{\mathbf{v}} f_{\theta,s,\zeta}(\mathbf{y})| \leq C_f |\mathbf{v}|! \prod_{j=1}^s \beta_j^{v_j}$ for all $\mathbf{v} \in \{0, 1, \dots, \alpha\}^s$ and for a $C_f \geq 1$ with the weights β_j given by $\beta_j = \theta a_j = \theta j^{-\zeta}$, $j = 1, \dots, s$. Additionally, for $s \rightarrow \infty$, 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 [6], we have that an interlaced polynomial lattice rule of order α with $N = b^m$ points (b prime, $m \geq 1$) and point set \mathcal{P}_N can be constructed such that

the QMC quadrature error fulfills

$$|\mathcal{I}[f_{\theta,s,\zeta}] - \mathcal{Q}_{\mathcal{P}_N}[f_{\theta,s,\zeta}]| \leq C(\alpha, \boldsymbol{\beta}, b, p) N^{-1/p}, \quad (19)$$

for a constant $C(\alpha, \boldsymbol{\beta}, b, p)$ independent of s and N . Convergence rates were computed with respect to a reference value of the integral $\mathcal{I}[f_{\theta,s,\zeta}]$ obtained with dimension-adaptive Smolyak quadrature with tolerance 10^{-14} . We also consider separable integrand functions, which, on account of their separability, trivially belong to the product weight class. They are given by

$$g_{\theta,s,\zeta}(\mathbf{y}) = \exp\left(\boldsymbol{\theta} \sum_{j=1}^s a_j y_j\right), \quad a_j = j^{-\zeta}, \quad (20)$$

and satisfy $\partial_{\mathbf{y}}^{\mathbf{v}} g(\mathbf{y}) = g(\mathbf{y}) \prod_{k=1}^s (\theta a_k)^{v_k}$. Under the assumption that there exists a constant $\tilde{C} > 0$ that is independent of s and such that $g(\mathbf{y}) \leq \tilde{C}$ for all $\mathbf{y} \in [0, 1]^s$, which holds here with $\tilde{C} = \exp(\boldsymbol{\theta} \sum_{j=1}^s j^{-\zeta})$, $\zeta > 1$, we have the bound $|\partial_{\mathbf{y}}^{\mathbf{v}} g_{\theta,s,\zeta}(\mathbf{y})| \leq \tilde{C} \prod_{j=1}^s \beta_j^{v_j}$, for all $\mathbf{v} \in \{0, 1, \dots, \alpha\}^s$ with the weights β_j given by $\beta_j = \theta a_j = \boldsymbol{\theta} j^{-\zeta}$ for $j = 1, \dots, s$. We have the following analytically given formula for the integral

$$\mathcal{I}[g_{\theta,s,\zeta}] = \prod_{j=1}^s \left[\frac{j^\zeta}{\boldsymbol{\theta}} \left(\exp(\boldsymbol{\theta} j^{-\zeta}) - 1 \right) \right] = \exp \left[\sum_{j=1}^s \log \left(\sum_{\mu=0}^{\infty} \frac{(\boldsymbol{\theta} j^{-\zeta})^\mu}{(\mu+1)!} \right) \right], \quad (21)$$

and have an error bound of the form (19), with a different value for $C(\alpha, \boldsymbol{\beta}, b, p)$.

5.2 Validation of Work Bound

The results below are based on an implementation of the CBC algorithm in the C++ programming language, and exploits shared-memory parallelism to reduce the computation time for large m and s . Fourier transforms were realized using the FFTW library, with shared-memory parallelization enabled. Timings were executed on a system with 48 Intel Xeon E5-2696 cores at 2.70 GHz, where at most 8 CPUs were used at a time. The timing results in Figure 1 show that the work bounds $\mathcal{O}(\alpha s N \log N + \alpha^2 s^2 N)$ for SPOD weights from [6, Thm. 3.1] and $\mathcal{O}(\alpha s N \log N)$ for product weights from [6, Thm. 3.2] are fulfilled in practice and seem to be tight. The work $\mathcal{O}(N \log N)$ 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 (18), 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. For base $b = 2$ in (4), it was recently shown in [20] that $C = 1$.

5.4 Higher-Order Convergence

As can be seen in Figures 3 and 4, the higher-order convergence rates proved in [6] can be observed in practice for the two classes of tested integrand functions. To generate the QMC rules used in Figures 3 and 4, the *ad hoc* value $C = 0.1$ was used. We also mention that for more general, non-affine, holomorphic parameter dependence of operators the same convergence rates and derivative bounds as in [6] have been recently established in [7]. The CBC constructions apply also to QMC rules for these (non affine-parametric) problems. The left subgraphs ($\zeta = 2$) show that higher values of the interlacing parameter α do not imply higher convergence rates, if the integrand does not exhibit sufficient sparsity as quantified by the norm (2). The right subgraphs ($\zeta = 4$) in Figures 3 and 4 show that the convergence rate is indeed dimension independent, but limited by the interlacing parameter $\alpha = 2$: the integrand function with $\zeta = 4$ affords higher rates than 2 for interlaced polynomial lattice rules with higher values of the interlacing parameter α .

The fast CBC constructions [15, 16], as adapted to higher order, interlaced polynomial lattice rules in [6], attain the asymptotic scalings for work and memory with respect to N and to integration dimension s already for moderate values of s and N . Theoretically predicted, dimension-independent convergence orders beyond first order were achieved with pruned generating vectors obtained with base $b = 2$ and Walsh constant $C = 1$. QMC rule performance was observed to be sensitive to overestimated values of the Walsh constant $C_{\alpha,b}$. The choice $b = 2$ and $C = 1$ with pruning of generating vectors, theoretically justified in [5] and [20], respectively, yielded satisfactory results for $\alpha = 2, 3, 4$ in up to $s = 1000$ dimensions.

Acknowledgements This work is supported by the Swiss National Science Foundation (SNF) under project number SNF149819 and by the European Research Council (ERC) under FP7 Grant AdG247277. Work of CS was performed in part while CS visited ICERM / Brown University in September 2014; the excellent ICERM working environment is warmly acknowledged.

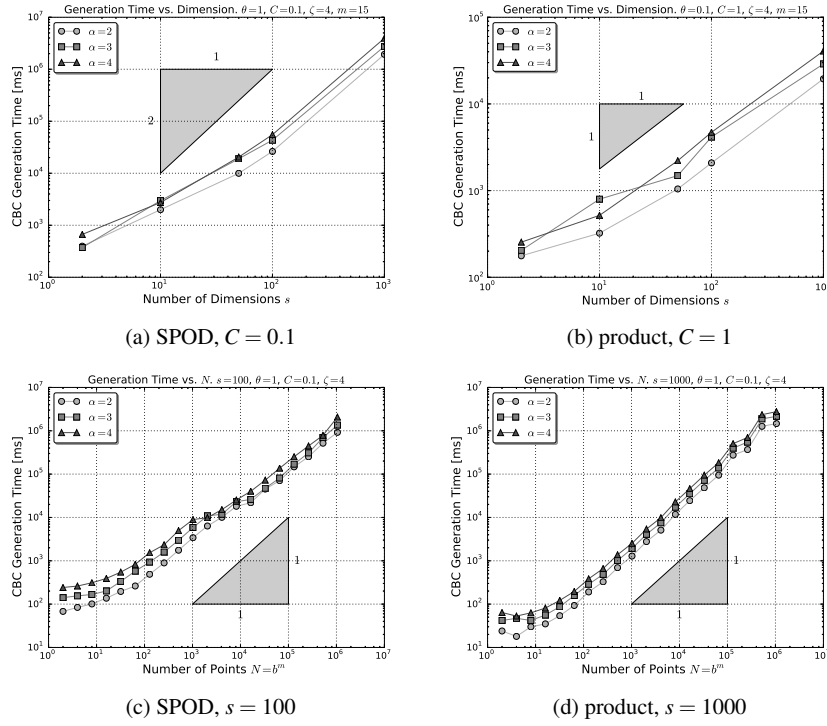


Fig. 1: CPU time required for the construction of generating vectors of varying order $\alpha = 2, 3, 4$ for product and SPOD weights with $\beta_j = \theta j^{-\zeta}$ vs. the dimension s in (a) and (b) and vs. the number of points $N = 2^m$ in (c) and (d).

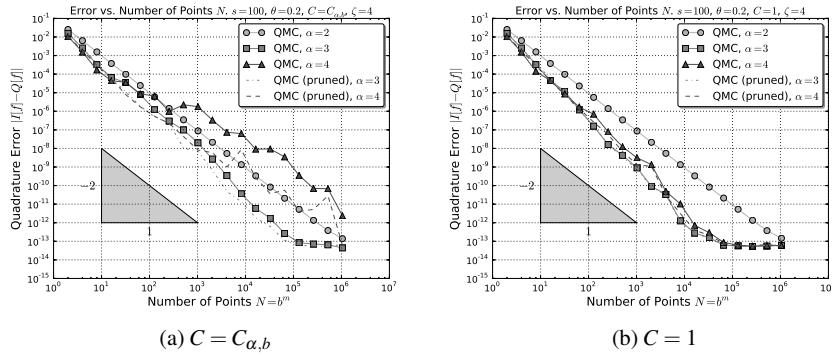


Fig. 2: Effect of pruning the generating vectors: convergence of QMC approximation for the SPOD integrand (18) with $\zeta = 4$, $s = 100$, base $b = 2$ and $\alpha = 2, 3, 4$, with and without pruning. Results (a) obtained with Walsh constant (4). In (b), the Walsh constant $C = 1$ and pruning are theoretically justified in [5] and [20], respectively.

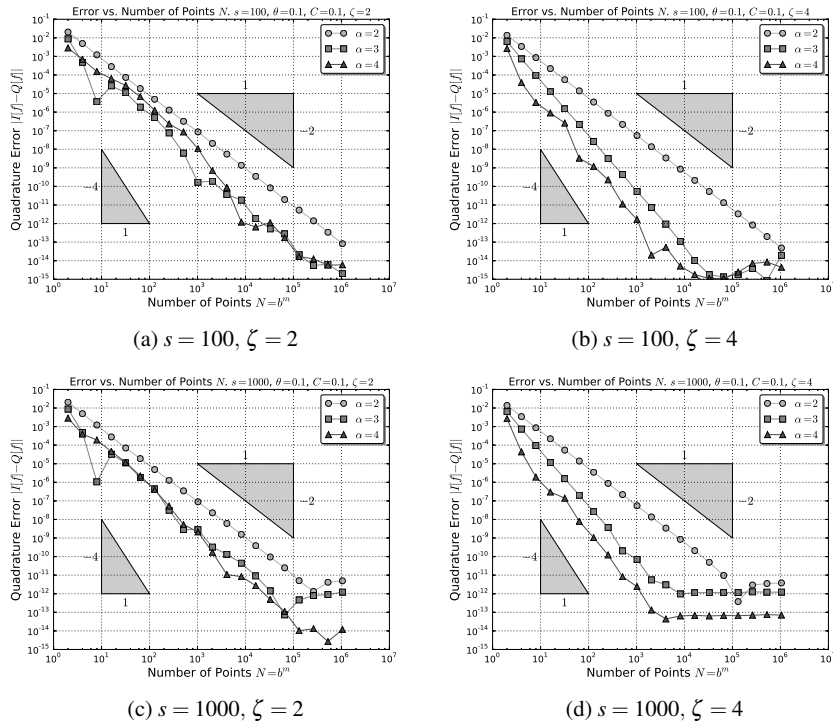


Fig. 3: Convergence of QMC approximation to (21) for the product weight integrand (20) in $s = 100, 1000$ dimensions with interlacing parameter $\alpha = 2, 3, 4$ with pruning.

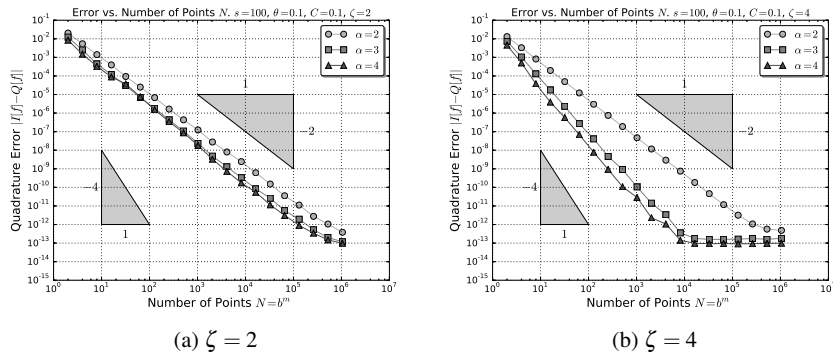


Fig. 4: Convergence of QMC approximation for the SPOD weight integrand (18) in $s = 100$ dimensions with interlacing parameter $\alpha = 2, 3, 4$ with pruning.

References

1. Dick, J.: Explicit constructions of quasi-Monte Carlo rules for the numerical integration of high-dimensional periodic functions. *SIAM J. Numer. Anal.* **45**(5), 2141–2176 (electronic) (2007). DOI 10.1137/060658916.
2. Dick, J.: Walsh spaces containing smooth functions and quasi-Monte Carlo rules of arbitrary high order. *SIAM J. Numer. Anal.* **46**(3), 1519–1553 (2008). DOI 10.1137/060666639.
3. Dick, J.: The decay of the Walsh coefficients of smooth functions. *Bull. Aust. Math. Soc.* **80**(3), 430–453 (2009). DOI 10.1017/S0004972709000392
4. Dick, J.: Random weights, robust lattice rules and the geometry of the *cbcrc* algorithm. *Numerische Mathematik* **122**(3), 443–467 (2012). DOI 10.1007/s00211-012-0469-5.
5. Dick, J., Kritzer, P.: On a projection-corrected component-by-component construction. *Journal of Complexity* (2015) DOI 10.1016/j.jco.2015.08.001.
6. Dick, J., Kuo, F. Y., Le Gia, Q. T., Nuyens, D., Schwab, C.: Higher order QMC Petrov-Galerkin discretization for affine parametric operator equations with random field inputs. *SIAM J. Numer. Anal.* **52**(6) (2014), pp. 2676–2702,
7. Dick, J., Le Gia, Q.T., Schwab, C.: Higher-order quasi-Monte Carlo integration for holomorphic, parametric operator equations. Tech. Rep. 2014-23, Seminar for Applied Mathematics, ETH Zürich (2014)
8. Dick, J., Pillichshammer, F.: *Digital nets and sequences*. Cambridge University Press, Cambridge (2010). DOI 10.1017/CBO9780511761188
9. Gantner, R. N.: Dissertation ETH Zürich (in preparation).
10. Goda, T.: Good interlaced polynomial lattice rules for numerical integration in weighted Walsh spaces. *J. Comput. Appl. Math.* **285**, 279–294 (2015). DOI 10.1016/j.cam.2015.02.041.
11. Goda, T., Dick, J.: Construction of interlaced scrambled polynomial lattice rules of arbitrary high order. *Found. Comput. Math.* (2015) DOI 10.1007/s10208-014-9226-8
12. 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
13. 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
14. 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
15. 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
16. 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
17. Rader, C.: Discrete Fourier transforms when the number of data samples is prime. *Proceedings of the IEEE* **3**(3), 1–2 (1968)
18. 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
19. 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
20. T. Yoshiki, Bounds on Walsh coefficients by dyadic difference and a new Koksma-Hlawka type inequality for Quasi-Monte Carlo integration. ArXiv:1504.03175 (2015).

Recent Research Reports

Nr.	Authors/Title
2014-15	Ch. Schwab 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. Schäfer $\backslash(\alpha)$ -Molecules
2014-17	A. Hildebrand and S. Mishra 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 Weak convergence rates of spectral Galerkin approximations for SPDEs with nonlinear diffusion coefficients
2014-19	J. Doelz and H. Harbrecht and Ch. Schwab Covariance regularity and H-matrix approximation for rough random fields
2014-20	P. Grohs and S. Hosseini Nonsmooth Trust Region Algorithms for Locally Lipschitz Functions on Riemannian Manifolds
2014-21	P. Grohs and A. Obermeier Optimal Adaptive Ridgelet Schemes for Linear Transport Equations
2014-22	S. Mishra and Ch. Schwab and J. Sukys 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 Higher order Quasi Monte Carlo integration for holomorphic, parametric operator equations
2014-24	C. Sanchez-Linares and M. de la Asuncion and M. Castro and S. Mishra and J. Šukys Multi-level Monte Carlo finite volume method for shallow water equations with uncertain parameters applied to landslides-generated tsunamis