ÖAW RICAM

Successive component search algorithms for good lattice points

Peter Kritzer

Johann Radon Institute for Computational and Applied Mathematics (RICAM)
Austrian Academy of Sciences
Linz, Austria

Joint work with
A. Ebert (KU Leuven)

On the frontiers of high dimensional integration, June 11, 2018

Research supported by the Austrian Science Fund, Project F5506-N26

SFB QMC

Introduction and Motivation

Consider integration of functions on $[0,1]^d$,

$$I_d(f) = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\,,$$

where $f \in \mathcal{H}$, and $\mathcal{H}$ is some Banach or Hilbert space.

Approximate I_d by a quasi-Monte Carlo (QMC) rule,

$$I_d(f) \approx Q_{N,d}(f) = \frac{1}{N}\sum_{k=0}^{N-1} f(\boldsymbol{x}_k),$$

where $\mathcal{P}_N = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\}$.

Both parameters d and N can be (very) large.

Worst case error in Banach space $\mathcal{H}$ with respect to $\mathcal{P}_N = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\}$:

$$e_{N,d}(\mathcal{H}, \mathcal{P}_N) := \sup_{f \in \mathcal{H}, \|f\| \leq 1} |I_d(f) - Q_{N,d}(f)| .$$

Need $\mathcal{P}_N$ that makes $e_{N,d}(\mathcal{H}, \mathcal{P}_N)$ small: How can we find it?

Function space:

Weighted Korobov space $(\mathcal{H}_{d,\alpha,\boldsymbol{\gamma}}, \|\cdot\|_{d,\alpha,\boldsymbol{\gamma}})$: space of continuous functions f, where

$$\|f\|^2_{d,\alpha,\boldsymbol{\gamma}} = \sum_{\boldsymbol{h}\in\mathbb{Z}^d} \rho_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h})\, |\widehat{f}(\boldsymbol{h})|^2,$$

where $\widehat{f}(\boldsymbol{h})$is the $\boldsymbol{h}$-th Fourier coefficient of f.

Here, $\rho_{\alpha,\boldsymbol{\gamma}}(\boldsymbol{h})$ moderates the decay of the Fourier coefficients, depending on:

- $\alpha > 1$: "smoothness parameter" (higher $\alpha \to$ smoother functions in $\mathcal{H}_{d,\alpha,\boldsymbol{\gamma}}$),
- $\boldsymbol{\gamma} = (\gamma_{\mathfrak{u}})_{\mathfrak{u}\subseteq\{1,\dots,d\}}$: coordinate weights.

Here:

$\mathcal{P}_N = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}\}$ is a lattice point set with generating vector

$$\boldsymbol{z} = (z_1, \ldots, z_d) \in \{1, \ldots, N-1\}^d.$$

Points of $\mathcal{P}_N$:

$$\boldsymbol{x}_n = (x_{n,1}, \ldots, x_{n,d})$$

with

$$x_{n,j} = \left\{ \frac{n z_j}{N} \right\},$$

where $\{t\} = t - \lfloor t \rfloor$.

Note: Given N and d, $\boldsymbol{z}$ fully determines the lattice point set.

Explicit formula for the (squared) worst-case error,

$$e^2_{N,d,\alpha,\gamma}(\boldsymbol{z}) = -1 + \frac{1}{N} \sum_{n=0}^{N-1} \prod_{j=1}^{d} \left(1 + \gamma_j \varphi_\alpha \left(\left\{\frac{n z_j}{N}\right\}\right)\right),$$

where $\varphi_\alpha\left(\frac{k}{N}\right)$ can be computed for all values of $k = 0, \ldots, N-1$.

The error formula is easy to implement.

Bounds on the worst-case error of lattice rules in the Korobov space with $\alpha = 2$ immediately yield bounds on the worst-case error of slightly modified lattice rules in certain Sobolev spaces.

- All that remains is to find "good" $\boldsymbol{z} \in \{1, \ldots, N-1\}^d$.
- Huge search space of size $(N-1)^d$. (e.g., $N = 10\,000$ and $d = 100$).
- Component by component (CBC) construction: greedy algorithm to construct z_j one at a time.
  Size of search space is $N - 1$ per component.
  (Almost) optimal convergence of error bounds.
- Fast CBC (Cools, Nuyens, 2006): computation cost of $\mathcal{O}(dN \log N)$.

The successive coordinate search (SCS) algorithm

Question: can one improve on the quality of the output vector of the CBC algorithm?

Ebert/Leövey/Nuyens (2016): successive coordinate search (SCS) algorithm.

Basic idea:

- Assume product weights.
- Begin with a start vector $\boldsymbol{z}^{(0)} = (z_1^{(0)}, \ldots, z_d^{(0)})$.
- Update components one after the other by minimizing worst-case error.
- Returned vector is at least as good as initial vector.

Algorithm 1 (SCS construction)

Let N be a prime. Let $\boldsymbol{z}^{(0)} = (z_1^{(0)}, \ldots, z_d^{(0)}) \in \{0, 1, \ldots, N-1\}^d$ be given.

- *For $s \in \{1, \ldots, d\}$ assume that $z_1, \ldots, z_{s-1}$ have already been found. Now choose $z_s \in \{1, \ldots, N-1\}$ such that*

$$e^2_{N,d,\alpha,\gamma}((z_1, \ldots, z_{s-1}, z_s, z^{(0)}_{s+1}, \ldots, z^{(0)}_d))$$

  is minimized as a function of z_s.
- *Increase s and repeat the second step until $\boldsymbol{z} = (z_1, \ldots, z_d)$ is found.*

Theorem 2 (Ebert/Leövey/Nuyens, 2016)

Let $\boldsymbol{z}^{(0)}$ be the initial vector in the SCS algorithm, and $\boldsymbol{z}$ the vector returned by the SCS algorithm. Then

$$e^2_{N,d,\alpha,\gamma}(\boldsymbol{z}) \leq e^2_{N,d,\alpha,\gamma}(\boldsymbol{z}^{(0)}).$$

Theorem 3 (Ebert/Leövey/Nuyens, 2016)

Set $\boldsymbol{z}^{(0)} := (0, 0, \ldots, 0)$ in the SCS algorithm.
Then the SCS algorithm yields the same result as the usual CBC algorithm. The SCS algorithm is thus a generalization of the CBC algorithm.

Fast implementation similar to fast CBC construction:

- Pre-computation with cost of $\mathcal{O}(dN)$,
- matrix-vector multiplication of same speed as in fast CBC algorithm,
- total cost of $\mathcal{O}(dN \log N)$.

The reduced fast SCS algorithm

Joint work with A. Ebert (2017/18):

Try to speed up SCS construction: Reduced fast SCS construction.

(**2015:** reduced fast CBC construction, together with Dick, Leobacher, Pillichshammer).

Idea: make search space smaller for later components of $\boldsymbol{z}$.

- Let N be a prime power, $N = b^m$, b prime, $m \in \mathbb{N}$.
- Let $w_1, \ldots, w_d, \ldots \in \mathbb{N}_0$ with $0 = w_1 \leq \cdots \leq w_d \leq \cdots$.
- Consider the sequence of reduced search spaces

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

- Note that

$$|\mathcal{Z}_{N,w_j}| := \begin{cases} b^{m-w_j-1}(b-1) & \text{if} \quad w_j < m, \\ 1 & \text{if} \quad w_j \geq m. \end{cases}$$

- write $Y_j := b^{w_j}$.

Algorithm 4 (Reduced SCS construction)

Let N be a prime power. Let $\boldsymbol{z}^{(0)} = (z_1^{(0)}, \ldots, z_d^{(0)}) \in \{0, 1, \ldots, N-1\}^d$ be given.

- *For $s \in \{1, \ldots, d\}$ assume that $z_1, \ldots, z_{s-1}$ have already been found. Now choose $z_s \in \mathcal{Z}_{N,w_s}$ such that*

$$e^2_{N,d,\alpha,\gamma}((Y_1 z_1, \ldots, Y_{s-1} z_{s-1}, Y_s z_s, z^{(0)}_{s+1}, \ldots, z^{(0)}_d))$$

  is minimized as a function of z_s.
- *Increase s and repeat the second step until $\boldsymbol{z} = (Y_1 z_1, \ldots, Y_d z_d)$ is found.*

Theorem 5 (Ebert/K., 2018)

Assume product weights and let $\boldsymbol{z} = (Y_1 z_1, \ldots, Y_d z_d)$ *be constructed by Algorithm 4. Then we have for all* $\delta \in (0, \frac{\alpha-1}{2}]$ *that*

$$e_{N,d,\alpha,\boldsymbol{\gamma}}(\boldsymbol{z}) \leq C_{d,\alpha,\boldsymbol{\gamma},\delta}\ N^{-\alpha/2+\delta},$$

where $C_{d,\alpha,\boldsymbol{\gamma},\delta}$ *is bounded independently of* d *if*

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

Theorem formulated for product weights, similar result holds for general weights.

Special case of reduced SCS construction:

Set $w_1 = w_2 = \cdots = w_d = 0$. Then the new result generalizes the previous SCS construction with respect to

- General weights instead of product weights,
- prime power N instead of prime N.

Reduced fast SCS construction:

- Assume product weights,
- Assume initial vector of the form $\boldsymbol{z}^{(0)} = (Y_1 z_1^{(0)}, \dots, Y_d z_d^{(0)})$ with $z_j^{(0)} \in \mathcal{Z}_{N,w_j}$.
- Use fast pre-computation (due to structure of $\boldsymbol{z}^{(0)}$) and fast matrix-vector multiplication (as in reduced fast CBC construction).
- Implementation with overall cost

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

where $d^* := \max\{j \in \mathbb{N} : w_j < m\}$.

Computation times

Computation times (in seconds):
unreduced (normal font) and reduced SCS (**bold font**)
$\alpha = 2, b = 2, \gamma_j = (0.7)^j, w_j = \lfloor 3 \log_b j \rfloor$

	$d = 50$	$d = 100$	$d = 500$	$d = 1000$	$d = 2000$
$m = 10$	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**

Intel Core i5-2400S CPU with 2.5GHz using Matlab.

Polynomial lattice rules:

- Polynomial lattice rules: similar to lattice rules, but integer arithmetic replaced by polynomial arithmetic over finite fields.
- Special cases of Niederreiter's (t, m, d)-nets, powerful QMC methods.
- All results shown above work analogously for polynomial lattice rules.

Thanks for your attention.

J. Dick, P. Kritzer. On a projection-corrected component-by-component construction. J. Complexity 32, 74–80, 2016.

J. Dick, P. Kritzer, G. Leobacher, F. Pillichshammer. *A reduced fast component-by-component construction of lattice points for integration in weighted spaces with fast decreasing weights*. J. Comput. Appl. Math. 276, 1–15, 2015.

A. Ebert, P. Kritzer. Constructing lattice points for numerical integration by a reduced fast successive coordinate search algorithm. Submitted, 2018.

A. Ebert, H. Leövey, D. Nuyens. *Successive Coordinate Search and Component-by-Component Construction of Rank-1 Lattice Rules*. To appear in: P. Glynn, A. Owen (eds.), *Monte Carlo and Quasi-Monte Carlo Methods 2016*, Springer, 2018.

H. Laimer. On combined component-by-component constructions of lattice point sets. J. Complexity 38, 22–30, 2017.

D. Nuyens, R. Cools. *Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces*. Math. Comp. 75, 903–920, 2006.