ÖAW AUSTRIAN ACADEMY OF SCIENCES

RICAM – JOHANN RADON INSTITUTE FOR COMPUTATIONAL AND APPLIED MATHEMATICS

The fast reduced QMC matrix-vector product

J. Dick, A. Ebert, L. Herrmann, P. Kritzer, M. Longo

RICAM-Report 2023-14

The fast reduced QMC matrix-vector product

Josef Dick, Adrian Ebert, Lukas Herrmann, Peter Kritzer, Marcello Longo

May 19, 2023

Abstract

We study the approximation of integrals of the form $\int_D f(\boldsymbol{x}^\top A)\,\mathrm{d}\mu(\boldsymbol{x})$, where A is a matrix, by quasi-Monte Carlo (QMC) rules $N^{-1}\sum_{k=0}^{N-1} f(\boldsymbol{x}_k^\top A)$. We are interested in cases where the main computational cost in the approximation arises from calculating the products $\boldsymbol{x}_k^\top A$. We design QMC rules for which the computation of $\boldsymbol{x}_k^\top A$, $k=0,1,\ldots,N-1$, can be done in a fast way, and for which the approximation error of the QMC rule is similar to the standard QMC error. We do not require that the matrix A has any particular structure.

Problems of this form arise in some important applications in statistics and uncertainty quantification. For instance, this approach can be used when approximating the expected value of some function with a multivariate normal random variable with some given covariance matrix, or when approximating the expected value of the solution of a PDE with random coefficients.

The speed-up of the computation time of our approach is sometimes better and sometimes worse than the fast QMC matrix-vector product from [Josef Dick, Frances Y. Kuo, Quoc T. Le Gia, and Christoph Schwab, Fast QMC Matrix-Vector Multiplication, SIAM J. Sci. Comput. 37 (2015), no. 3, A1436–A1450]. As in that paper, our approach applies to lattice point sets and polynomial lattice point sets, but also applies to digital nets (we are currently not aware of any approach which allows one to apply the fast QMC matrix-vector paper from the aforementioned paper of Dick, Kuo, Le Gia, and Schwab to digital nets).

The method in this paper does not make use of the fast Fourier transform, instead we use repeated values in the quadrature points to derive a significant reduction in the computation time. Such a situation naturally arises from the reduced CBC construction of lattice rules and polynomial lattice rules. The reduced CBC construction has been shown to reduce the computation time for the CBC construction. Here we show that it can additionally be used to also reduce the computation time of the underlying QMC rule. One advantage of the present approach is that it can be combined with random (digital) shifts, whereas this does not apply to the fast QMC matrix-vector product from the earlier paper of Dick, Kuo, Le Gia, and Schwab.

Keywords: Matrix-vector multiplication, quasi-Monte Carlo, high-dimensional integration, lattice rules, polynomial lattice rules, digital nets, PDEs with random coefficients. **2020 MSC:** 65C05, 65D30, 41A55, 11K38.

1 Introduction and problem setting

We are interested in approximating integrals of the form

$$\int_D f(\boldsymbol{x}^\top A)\,\mathrm{d}\mu(\boldsymbol{x}), \tag{1}$$

for a domain $D \subseteq \mathbb{R}^s$, an $s \times \tau$-matrix $A \in \mathbb{R}^{s\times\tau}$, and a function $f : D \to \mathbb{R}$, by quasi-Monte Carlo (QMC) integration rules of the form

$$Q_N(f) = \frac{1}{N} \sum_{k=0}^{N-1} f(\boldsymbol{x}_k^\top A), \tag{2}$$

where we use deterministic cubature points $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1} \in D$. We write $\boldsymbol{x}_k = (x_{1,k}, x_{2,k}, \ldots, x_{s,k})^\top$ for $0 \le k \le N-1$. In most instances, $D = [0,1]^s$ and the measure μ is the Lebesgue measure (or $D = \mathbb{R}^s$ and μ is the measure corresponding to the normal distribution).

Furthermore, define the $N \times s$-matrix

$$X = \begin{pmatrix} \boldsymbol{x}_0^\top \\ \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_{N-1}^\top \end{pmatrix} \in \mathbb{R}^{N\times s}, \tag{3}$$

whose N rows consist of the different cubature nodes. We are interested in situations where the main computational cost of computing (2) arises from the vector-matrix multiplication $\boldsymbol{x}_k^\top A$ for all N points, i.e., we need to compute XA, which requires $\mathcal{O}(Ns\,t)$ operations. Let $A = (\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_\tau)$, where $\boldsymbol{A}_i \in \mathbb{R}^s$ is the i-th column vector of A. The main idea is to construct QMC rules for which the matrix X given in (3) has some structure such that QMC matrix-vector product $X\boldsymbol{A}_i$ can be computed very efficiently and the integration error of the underlying QMC rule has similar properties as for other QMC rules. Note that our approach works for any matrix A as we do not use any structure of the matrix A.

To motivate the problem addressed in this paper, note that such computational problems arise naturally in certain settings. For instance, consider approximating the expected value

$$\mathbb{E}(f) = \int_{\mathbb{R}^s} f(\boldsymbol{y}^\top) \frac{\exp\left(-\frac{1}{2}\boldsymbol{y}^\top \Sigma^{-1} \boldsymbol{y}\right)}{\sqrt{(2\pi)^s \det(\Sigma)}}\, \mathrm{d}\boldsymbol{y},$$

where Σ is symmetric and positive definite. Using the substitution $\boldsymbol{y} = A^\top \boldsymbol{x}$, where A factorizes Σ, i.e. $\Sigma = A^\top A$, we arrive at the integral

$$\mathbb{E}(f) = \int_{\mathbb{R}^s} f(\boldsymbol{x}^\top A) \underbrace{\frac{\exp\left(-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{x}\right)}{(2\pi)^{s/2}}\, \mathrm{d}\boldsymbol{x}}_{=:\,\mathrm{d}\mu(\boldsymbol{x})}.$$

Such problems arise for instance in statistics when computing expected values with respect to a normal distribution, and in mathematical finance, e.g., for pricing financial products whose payoff depends on a basket of assets.

Another setting where such problems arise naturally comes from PDEs with random coefficients in the context of uncertainty quantification (see for instance [12] for more details). Without stating all the details here, the main computational cost in this context comes from computing

$$D_k = \sum_{j=1}^{s} x_{j,k} C_j, \quad \text{for } k = 0, 1, \ldots, N-1, \tag{4}$$

where $C_j \in \mathbb{R}^{M\times M}$ are matrices (whose size depends on s and N). Let $C_j = (c_{j,u,v})_{1\le u,v\le M}$ and define the column vectors $\boldsymbol{c}_{u,v} = (c_{1,u,v}, c_{2,u,v}, \ldots, c_{s,u,v})^\top \in \mathbb{R}^s$, for $1 \le u, v \le M$. Then we can compute the matrices given by (4) by computing

$$X\boldsymbol{c}_{u,v}, \quad \text{for } 1 \le u, v \le M. \tag{5}$$

In this approach we do not compute the matrices in (4) for each k separately, hence this approach requires us to store the results of (5) first.

It was shown in [4] that when using particular types of QMC rules, such as (polynomial) lattice rules or Korobov rules, the cost to evaluate $Q_N(f)$, as given in (2), can be reduced to only $\mathcal{O}(\tau\, N \log N)$ operations provided that $\log N \ll s$. This drastic reduction in computational cost is achieved by a fast matrix-matrix multiplication exploiting the fact that for the chosen point sets the matrix X can be re-ordered to be of circulant structure. The fast multiplication is then realized by the use of the fast Fourier transformation (FFT).

Here, we will explore a different method which can also drastically reduce the computation cost of evaluating $Q_N(f)$, as given in (2). The reduction in computational complexity is achieved by using point sets which possess a certain repetitiveness in their components. In particular, the number of different values of the components $x_{k,j}$ (for $0 \le k \le N-1$) is in general smaller than N and decreases when $j \in \{1, \ldots, s\}$ increases. As a particular type of such QMC point sets, we will consider (polynomial) lattice point sets that have been obtained by the so-called reduced CBC construction as in [2], and we will also consider similarly reduced versions of digital nets obtained from digital sequences such as Sobol' or Niederreiter sequences. The corresponding QMC point sets will henceforth be called reduced (polynomial) lattice point sets or reduced digital nets.

The idea of our approach, which will be made more precise in the following sections, works as follows.

Assume that we have N samples of the form $(x_{1,k}, x_{2,k}, \ldots, x_{s,k})$, $0 \le k \le N-1$. We reduce the number of different values by choosing the number of samples differently for each coordinate, say N_j for the j-th coordinate, where N_j divides N_{j-1}. E.g., if $N_1 = 4$, $N_2 = 2$, and $N_3 = 1$, then we generate the points

$$(y_{1,0}, y_{2,0}, y_{3,0}), (y_{1,1}, y_{2,1}, y_{3,0}), (y_{1,2}, y_{2,0}, y_{3,0}), (y_{1,3}, y_{2,1}, y_{3,0}). \tag{6}$$

Here, there are 4 different values for the first coordinate, 2 different values for the second coordinate, and the values for the last coordinate are all the same.

What is the advantage of this construction? The advantage can be seen when we compute $\boldsymbol{x}^\top A$. Let $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_s$ denote the rows of A. If all coordinates are different, we need $\mathcal{O}(Ns)$ operations. For instance, in the example above we have 4 points in the 3-dimensional space, so we need to compute

$$x_{1,k}\boldsymbol{a}_1 + x_{2,k}\boldsymbol{a}_2 + x_{3,k}\boldsymbol{a}_3, \quad \text{for} \quad k \in \{1,2,3,4\}.$$

However, if we use the points (6) then we only need to compute

$$y_{1,k}\boldsymbol{a}_1 + y_{2,\lfloor k/2\rfloor}\boldsymbol{a}_2 + y_{3,0}\boldsymbol{a}_3, \quad \text{for} \quad k \in \{1,2,3,4\}.$$

The last computation can be done recursively, by first computing $y_{3,0}\boldsymbol{a}_3$, then $y_{2,0}\boldsymbol{a}_2 + y_{3,0}\boldsymbol{a}_3$ and $y_{3,1}\boldsymbol{a}_2 + y_{3,0}\boldsymbol{a}_3$, and then finally the remaining vectors. By storing and reusing these

intermediate results, we only compute $y_{3,0}\boldsymbol{a}_3$, and $y_{2,0}\boldsymbol{a}_2 + y_{3,0}\boldsymbol{a}_3$ and $y_{2,1}\boldsymbol{a}_2 + y_{3,0}\boldsymbol{a}_3$ once (rather than recomputing the same result as in the straightforward computation).

By applying this idea in the general case, we obtain a similar cost saving as for the fast QMC matrix-vector product in [4]. However, the present method behaves differently in some situations which can be beneficial. One advantage is that it allows us to use random shifts, which is not possible for the fast QMC matrix-vector product.

Before we proceed, we would like to introduce some notation. We will write $\mathbb{Z}$ to denote the set of integers, $\mathbb{Z}_*$ to denote the set of integers excluding 0, $\mathbb{N}$ to denote the positive integers, and $\mathbb{N}_0$ to denote the nonnegative integers. Furthermore, we write $[s]$ to denote the index set $\{1, \ldots, s\}$. To denote sets of components we use fraktur font, e.g., $\mathfrak{u} \subseteq [s]$. For a vector $\boldsymbol{x} = (x_1, \ldots, x_s) \in [0,1]^s$ and for $\mathfrak{u} \subseteq [s]$, we write $\boldsymbol{x}_\mathfrak{u} = (x_j)_{j\in\mathfrak{u}} \in [0,1]^{|\mathfrak{u}|}$ and $(\boldsymbol{x}_\mathfrak{u}, \boldsymbol{0}) \in [0,1]^s$ for the vector $(y_1, \ldots, y_s)$ with $y_j = x_j$ if $j \in \mathfrak{u}$ and $y_j = 0$ if $j \notin \mathfrak{u}$. For integer vectors $\boldsymbol{h} \in \mathbb{Z}^s$, and $\mathfrak{u} \subseteq [s]$, we analogously write $\boldsymbol{h}_\mathfrak{u}$ to denote the projection of $\boldsymbol{h}$ onto those components with indices in $\mathfrak{u}$.

The rest of the paper is structured as follows. Below we introduce lattice rules and polynomial lattice rules and the relevant function spaces. In Section 1.3 we state the relevant results on the convergence of the reduced lattice rules. In Section 2 we outline how to use reduced rules for computing matrix products efficiently. In Section 3 we discuss a version of the fast reduced QMC matrix-vector multiplication for digital nets and prove a bound on the weighted discrepancy. In Section 4 we explain how these ideas can also be applied to the plain Monte Carlo algorithm. Numerical experiments in Section 5 conclude the paper.

1.1 Lattice point sets and polynomial lattice point sets

In this section, we would like to give the definitions of the classes of QMC point sets considered in this paper.

We start with (rank-1) lattice point sets. For further information, we refer to, e.g., [3, 5, 13, 15] and the references therein.

For a natural number $N \in \mathbb{N}$ and a vector $\boldsymbol{z} \in \{1, 2, \ldots, N-1\}^s$, a lattice point set consists of points $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1}$ of the form

$$\boldsymbol{x}_k = \left\{ \frac{k}{N} \boldsymbol{z} \right\} \quad \text{for} \quad k = 0, 1, \ldots, N-1.$$

Here, for real numbers $y \geq 0$ we write $\{y\} = y - \lfloor y \rfloor$ for the fractional part of y. For vectors $\boldsymbol{y}$ we apply $\{\cdot\}$ component-wise.

In this paper, we assume that the number of points N is a prime power, i.e., $N = b^m$, with prime b and $m \in \mathbb{N}$.

The second class of point sets considered here are so-called polynomial lattice point sets, whose definition is similar to that of lattice point sets, but based on arithmetic over finite fields instead of integer arithmetic. To introduce them, let b again be a prime, and denote by $\mathbb{F}_b$ the finite field with b elements and by $\mathbb{F}_b[x]$ the set of all polynomials in x with coefficients in $\mathbb{F}_b$. We will use a special instance of polynomial lattice point sets over $\mathbb{F}_b$. For a prime power $N = b^m$ and $\boldsymbol{g} = (g_1, \ldots, g_s) \in (\mathbb{F}_b[x])^s$, a polynomial lattice point

set consists of N points $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1}$ of the form

$$\boldsymbol{x}_k := \left(\nu\left(\frac{k(x)\ g_1(x)}{x^m}\right), \ldots, \nu\left(\frac{k(x)\ g_s(x)}{x^m}\right)\right) \quad \text{for } k \in \mathbb{F}_b[x] \text{ with } \deg(k) < m,$$

where for $f \in \mathbb{F}_b[x]$, $f(x) = a_0 + a_1 x + \cdots + a_r x^r$, with $\deg(f) = r$, the map ν is given by

$$\nu\left(\frac{f(x)}{x^m}\right) := \frac{a_{\min(r,m-1)}}{b^{m-\min(r,m-1)}} + \cdots + \frac{a_1}{b^{m-1}} + \frac{a_0}{b^m} \in [0,1).$$

Note that $\nu(f(x)/x^m) = \nu((f(x) \pmod{x^m})/x^m)$. We refer to [7, Chapter 10] for further information on polynomial lattice point sets.

Lattice point sets are used in QMC rules referred to as lattice rules, and analogously for polynomial lattice point sets.

1.2 Korobov spaces and related Sobolev spaces

As pointed out above, lattice point sets are commonly used as node sets in lattice rules, and they are frequently studied in the context of numerical integration of functions in Korobov spaces and certain Sobolev spaces, which we would like to describe in the present section. Let us consider first a weighted Korobov space with general weights as studied in [8, 14].

In several applications, we may have the situation that different groups of variables have different importance, and this can also be reflected in the function spaces under consideration. Indeed, the importance of the different components or groups of components of the functions in the Korobov space to be defined is specified by a set of positive real numbers $\boldsymbol{\gamma} = \{\gamma_{\mathfrak{u}}\}_{\mathfrak{u} \subseteq [s]}$, where we may assume that $\gamma_\emptyset = 1$. In this context, larger values of $\gamma_{\mathfrak{u}}$ indicate that the group of variables corresponding to the index set $\mathfrak{u}$ has relatively stronger influence on the computational problem, whereas smaller values of $\gamma_{\mathfrak{u}}$ mean the opposite.

The smoothness of the functions in the space is described by a parameter $\alpha > 1/2$.

Product weights are a common special case of the weights $\boldsymbol{\gamma}$ where $\gamma_{\mathfrak{u}} = \prod_{j \in \mathfrak{u}} \gamma_j$ for $u \subseteq [s]$ and where $(\gamma_j)_{j=1,2,\ldots,s}$ is a sequence of positive real numbers.

The weighted Korobov space, denoted by $\mathcal{H}(K_{s,\alpha,\boldsymbol{\gamma}})$, is a reproducing kernel Hilbert space with kernel function

$$\begin{aligned} K_{s,\alpha,\boldsymbol{\gamma}}(\boldsymbol{x}, \boldsymbol{y}) &= 1 + \sum_{\emptyset \neq \mathfrak{u} \subseteq [s]} \gamma_{\mathfrak{u}} \prod_{j \in \mathfrak{u}} \left(\sum_{h \in \mathbb{Z}_*} \frac{\exp(2\pi \mathrm{i} h(x_j - y_j))}{|h|^{2\alpha}} \right) \\ &= 1 + \sum_{\emptyset \neq \mathfrak{u} \subseteq [s]} \gamma_{\mathfrak{u}} \sum_{\boldsymbol{h}_{\mathfrak{u}} \in \mathbb{Z}_*^{|\mathfrak{u}|}} \frac{\exp(2\pi \mathrm{i} \boldsymbol{h}_{\mathfrak{u}} \cdot (\boldsymbol{x}_{\mathfrak{u}} - \boldsymbol{y}_{\mathfrak{u}}))}{\prod_{j \in \mathfrak{u}} |h_j|^{2\alpha}}. \end{aligned}$$

The corresponding inner product is

$$\langle f, g \rangle_{K_{s,\alpha,\boldsymbol{\gamma}}} = \sum_{\mathfrak{u} \subseteq [s]} \gamma_{\mathfrak{u}}^{-1} \sum_{\boldsymbol{h}_{\mathfrak{u}} \in \mathbb{Z}_*^{|\mathfrak{u}|}} \left(\prod_{j \in \mathfrak{u}} |h_j|^{2\alpha} \right) \widehat{f}((\boldsymbol{h}_{\mathfrak{u}}, \boldsymbol{0})) \overline{\widehat{g}((\boldsymbol{h}_{\mathfrak{u}}, \boldsymbol{0}))},$$

where $\widehat{f}(\boldsymbol{h}) = \int_{[0,1]^s} f(\boldsymbol{t}) \exp(-2\pi \mathrm{i} \boldsymbol{h} \cdot \boldsymbol{t}) \, \mathrm{d}\boldsymbol{t}$ is the $\boldsymbol{h}$-th Fourier coefficient of f. For $\mathfrak{u} = \emptyset$, the empty sum is defined as $\widehat{f}(\boldsymbol{0})\overline{\widehat{g}(\boldsymbol{0})}$.

For $h \in \mathbb{Z}_*$, we define $\rho_\alpha(h) = |h|^{-2\alpha}$, and for $\boldsymbol{h} = (h_1, \ldots, h_s) \in \mathbb{Z}_*^s$ let $\rho_\alpha(\boldsymbol{h}) = \prod_{j=1}^s \rho_\alpha(h_j)$.

It is known (see, e.g., [8]) that the squared worst-case error of a lattice rule generated by a vector $\boldsymbol{z} \in \mathbb{Z}^s$ in the weighted Korobov space $\mathcal{H}(K_{s,\alpha,\boldsymbol{\gamma}})$ is given by

$$e^2_{N,s,\boldsymbol{\gamma}}(\boldsymbol{z}) = \sum_{\emptyset \ne \mathfrak{u} \subseteq [s]} \gamma_\mathfrak{u} \sum_{\boldsymbol{h}_\mathfrak{u} \in \mathcal{D}_\mathfrak{u}} \rho_\alpha(\boldsymbol{h}_\mathfrak{u}), \tag{7}$$

where

$$\mathcal{D}_\mathfrak{u} = \mathcal{D}_\mathfrak{u}(\boldsymbol{z}) := \left\{ \boldsymbol{h}_\mathfrak{u} \in \mathbb{Z}_*^{|\mathfrak{u}|} \ : \ \boldsymbol{h}_\mathfrak{u} \cdot \boldsymbol{z}_\mathfrak{u} \equiv 0 \ (\mathrm{mod}\, N) \right\}$$

is called the dual lattice of the lattice generated by $\boldsymbol{z}$.

The worst-case error of lattice rules in a Korobov space can be related to the worst-case error in certain Sobolev spaces. Indeed, consider a tensor product Sobolev space $\mathcal{H}^{\mathrm{sob}}_{s,\boldsymbol{\gamma}}$ of absolutely continuous functions whose mixed partial derivatives of order 1 in each variable are square integrable, with norm (see [10])

$$\|f\|_{\mathcal{H}^{\mathrm{sob}}_{s,\boldsymbol{\gamma}}} = \left(\sum_{\mathfrak{u} \subseteq [s]} \gamma_\mathfrak{u}^{-1} \int_{[0,1]^{|\mathfrak{u}|}} \left(\int_{[0,1]^{s-|\mathfrak{u}|}} \frac{\partial^{|\mathfrak{u}|}}{\partial \boldsymbol{x}_\mathfrak{u}} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}_{[s]\setminus\mathfrak{u}} \right)^2 \mathrm{d}\boldsymbol{x}_\mathfrak{u} \right)^{1/2},$$

where $\partial^{|\mathfrak{u}|} f / \partial \boldsymbol{x}_\mathfrak{u}$ denotes the mixed partial derivative with respect to all variables $j \in \mathfrak{u}$. As pointed out in [5, Section 5], the root mean square worst-case error $\widehat{e}_{N,s,\boldsymbol{\gamma}}$ for QMC integration in $\mathcal{H}^{\mathrm{sob}}_{s,\boldsymbol{\gamma}}$ using randomly shifted lattice rules $(1/N)\sum_{k=0}^{N-1} f\left(\left\{\frac{k}{N}\boldsymbol{z} + \boldsymbol{\Delta}\right\}\right)$, i.e.,

$$\widehat{e}_{N,s,\boldsymbol{\gamma}}(\boldsymbol{z}) = \left(\int_{[0,1]^s} e^2_{N,s,\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{\Delta}) \, \mathrm{d}\boldsymbol{\Delta} \right)^{1/2},$$

where $e_{N,s,\boldsymbol{\gamma}}(\boldsymbol{z}, \boldsymbol{\Delta})$ is the worst-case error of QMC integration in $\mathcal{H}^{\mathrm{sob}}_{s,\boldsymbol{\gamma}}$ using a shifted integration lattice, is essentially the same as the worst-case error $e^{(1,\mathrm{kor})}_{N,s,\boldsymbol{\gamma}}$ in the weighted Korobov space $\mathcal{H}(K_{s,1,\boldsymbol{\gamma}})$ using the unshifted version of the lattice rules. In fact, we have

$$\widehat{e}_{N,s,2\pi^2\boldsymbol{\gamma}}(\boldsymbol{z}) = e^{(1,\mathrm{kor})}_{N,s,\boldsymbol{\gamma}}(\boldsymbol{z}), \tag{8}$$

where $2\pi^2\boldsymbol{\gamma}$ denotes the weights $((2\pi^2)^{|\mathfrak{u}|}\gamma_\mathfrak{u})_{\emptyset \ne \mathfrak{u} \subseteq [s]}$. For a connection to the so-called anchored Sobolev space see, e.g., [11, Section 4].

In a slightly different setting, the random shift can be replaced by the tent transformation $\phi(x) = 1 - |1 - 2x|$ in each variable. For a vector $\boldsymbol{x} \in [0,1]^s$ let $\phi(\boldsymbol{x})$ be defined componentwise. Let $\widetilde{e}_{N,s,\boldsymbol{\gamma}}(\boldsymbol{z})$ be the worst-case error in the unanchored weighted Sobolev space $\mathcal{H}^{\mathrm{sob}}_{s,\boldsymbol{\gamma}}$ using the QMC rule $(1/N)\sum_{k=0}^{N-1} f\left(\phi\left(\left\{\frac{k}{N}\boldsymbol{z}\right\}\right)\right)$. Then it is known due to [6] and [1] that

$$\widetilde{e}_{N,s,\pi^2\boldsymbol{\gamma}}(\boldsymbol{z}) \le e^{(1,\mathrm{kor})}_{N,s,\boldsymbol{\gamma}}(\boldsymbol{z}), \tag{9}$$

where $\pi^2\boldsymbol{\gamma} = (\pi^{2|\mathfrak{u}|}\gamma_\mathfrak{u})_{\emptyset \ne \mathfrak{u} \subseteq [s]}$, and that the CBC construction with the quality criterion given by the worst-case error in the Korobov space $\mathcal{H}(K_{s,1,\boldsymbol{\gamma}})$ can be used to construct

tent-transformed lattice rules which achieve the almost optimal convergence order in the space $\mathcal{H}^{\mathrm{sob}}_{s,\pi^2\boldsymbol{\gamma}}$ under appropriate conditions on the weights $\boldsymbol{\gamma}$ (see [1, Corollary 1]). Hence we also have a direct connection between integration in the Korobov space using lattice rules and integration in the unanchored Sobolev space using tent-transformed lattice rules.

Thus, results shown for the integration error in the Korobov space can, by a few simple modifications, be carried over to results that hold for anchored and unanchored Sobolev spaces, respectively, by using Equations (8) and (9).

1.3 Reduced (polynomial) lattice point sets

In [2], the authors introduced so-called reduced lattice point sets and reduced polynomial lattice point sets. The original motivation for these concepts was to make search algorithms for excellent QMC rules faster for situations where the dependence of a high-dimensional integration problem on its variable j decreases fast as the index j increases. Such a situation might occur in various applications and is modelled by assuming that the weights in the weighted spaces, such as those introduced in Section 1.2, decay at a certain speed.

The "reduction" in the search for good lattice point sets is achieved by shrinking the sizes of the sets that the different components of the generating vector $\boldsymbol{z}$ are chosen from. In the present paper, we will make use of the same idea, but with a different aim, namely that of increasing the speed of computing the matrix product XA, as outlined above.

Recall that we assume N to be a prime power, $N = b^m$. A reduced rank-1 lattice point set is obtained by introducing an integer sequence $\boldsymbol{w} = (w_j)_{j=1}^s \in \mathbb{N}_0^s$ with $0 = w_1 \le w_2 \le \cdots \le w_s$. We will refer to the integers w_j as reduction indices. Additionally, for integer $w \ge 0$, we introduce the set

$$\mathbb{U}_{b^{m-w}} := \begin{cases} \{z \in \{1,2,\ldots,b^{m-w}-1\} : \gcd(z,b) = 1\} & \text{if } w < m, \\ \{1\} & \text{if } w \ge m. \end{cases}$$

Note that $\mathbb{U}_{b^{m-w}}$ is the group of units of integers modulo b^{m-w} for $w < m$, and in this case the cardinality of the set $\mathbb{U}_{b^{m-w}}$ equals $(b-1)b^{m-w-1}$. For the given sequence $\boldsymbol{w}$ we then define s^* as $s^* := \max\{j \in \mathbb{N}\colon w_j < m\}$.

The generating vector $\boldsymbol{z} \in \mathbb{Z}^s$ of a reduced lattice rule as in [2] is then of the form

$$\boldsymbol{z} = (b^{w_1} z_1, b^{w_2} z_2, \ldots, b^{w_s} z_s) = (z_1, b^{w_2} z_2, \ldots, b^{w_s} z_s),$$

where $z_j \in \mathbb{U}_{b^{m-w_j}}$ for all $j = 1,\ldots,s$. Note that for $j > s^*$ we have $w_j \ge m$ and $z_j = 1$. In this case the corresponding components of $\boldsymbol{z}$ are multiples of N. The resulting b^m points of the reduced lattice point set are given by

$$\begin{aligned} \boldsymbol{x}_k &= \left(\left\{ \frac{k z_1 b^{w_1}}{N} \right\}, \left\{ \frac{k z_2 b^{w_2}}{N} \right\}, \ldots, \left\{ \frac{k z_s b^{w_s}}{N} \right\} \right) \\ &= \left(\frac{k z_1 \bmod b^{m-\min(w_1,m)}}{b^{m-\min(w_1,m)}}, \frac{k z_2 \bmod b^{m-\min(w_2,m)}}{b^{m-\min(w_2,m)}}, \ldots, \frac{k z_s \bmod b^{m-\min(w_s,m)}}{b^{m-\min(w_s,m)}} \right) \end{aligned}$$

with $k = 0,1,\ldots,N-1$. Therefore it is obvious that the components $x_{k,j}$, $k = 0,1,\ldots,N-1$ belong to the set $\{0, 1/b^{m-\min(w_j,m)}, \ldots, (b^{m-\min(w_j,m)}-1)/b^{m-\min(w_j,m)}\}$ and each of the values is attained exactly $b^{\min(w_j,m)}$ times for all $j = 1,\ldots,s$. In particular, all $x_{k,j}$ equal 0 for $j > s^*$.

Regarding the performance of reduced lattice rules for numerical integration in the Korobov space $\mathcal{H}(K_{s,\alpha,\gamma})$, the following result was shown in [2]. For a proof of this result and further background information, we refer to the original paper [2].

Theorem 1. *Let $\boldsymbol{w} = (w_j)_{j=1}^s \in \mathbb{N}_0^s$ be a sequence of reduction indices, let $\alpha > 1/2$, and consider the Korobov space $\mathcal{H}(K_{s,\alpha,\gamma})$. Using a computer search algorithm, one can construct a generating vector $\boldsymbol{z} = (z_1 b^{w_1}, \dots, z_s b^{w_s}) \in \mathbb{Z}^s$ such that, for any $d \in [s]$ and any $\lambda \in (1/(2\alpha), 1]$, the following estimate on the squared worst-case error of integration in $\mathcal{H}(K_{d,\alpha,\gamma})$ holds.*

$$e^2_{N,s,\gamma}((z_1 b^{w_1}, \dots, z_d b^{w_d})) \le \left(\sum_{\emptyset \ne \mathfrak{u} \subseteq [d]} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(2\alpha\lambda))^{|\mathfrak{u}|}}{b^{\max\{0, m - \max_{j \in \mathfrak{u}} w_j\}}} \right)^{\frac{1}{\lambda}}.$$

Let us briefly illustrate the motivation for introducing the numbers $w_1, w_2, \dots, w_s$. Assume we have product weights $\gamma_1 \ge \gamma_2 \ge \cdots \ge \gamma_s > 0$. We have

$$\sum_{\emptyset \ne \mathfrak{u} \subseteq [d]} \gamma_{\mathfrak{u}}^{\lambda} \frac{2(2\zeta(2\alpha\lambda))^{|\mathfrak{u}|}}{b^{\max\{0, m - \max_{j \in \mathfrak{u}} w_j\}}} \le b^{-m} \left(-1 + 2 \prod_{j=1}^{d} \left(1 + \gamma_j^{\lambda} 2\zeta(2\alpha\lambda) b^{\min\{m, w_j\}} \right) \right).$$

Further assume that we want to have a bound independent of the dimension. In the non-reduced (classical) case we have $w_1 = w_2 = \cdots = w_s = 0$ and hence

$$\prod_{j=1}^{d} \left(1 + \gamma_j^{\lambda} 2\zeta(2\alpha\lambda) \right) = \exp\left(\sum_{j=1}^{d} \log\left(1 + \gamma_j^{\lambda} 2\zeta(2\alpha\lambda) \right) \right) \le \exp\left(2\zeta(2\alpha\lambda) \sum_{j=1}^{d} \gamma_j^{\lambda} \right),$$

where we used $\log(1+z) \le z$ for $z \ge 0$. If $\sum_{j=1}^{\infty} \gamma_j^{\lambda} < \infty$, we get a bound which is independent of the dimension d.

For illustration, say $\gamma_j^{1/(2\alpha)} = j^{-4}$, then the infinite sum is finite and we get a bound independent of the dimension. However, a significantly slower converging sequence would still be enough to give us a bound independent of the dimension. So if we introduce $w_1 \le w_2 \le \cdots$, where $w_j = \log_b j^2$ for instance, then we still have

$$\sum_{j=1}^{\infty} \gamma_j^{\lambda} b^{w_j} < \infty.$$

In [2] we have shown how the w_j can be used to reduce the construction cost of the CBC construction by reducing the size of the search space from b^m to $b^{\max\{0, m - w_j\}}$ in component j. In this paper we show that the w_j can also be used to reduce the computation cost of computing XA, where the rows of X are the lattice points of a reduced lattice rule. The speed-up which can be achieved this way will depend on the weights $\{\gamma_{\mathfrak{u}}\}_{\mathfrak{u} \subseteq [s]}$ (and the $w_1 \ge w_2 \ge \cdots$). This is different from the fast QMC matrix-vector product in [4], which works independently of the weights and does not influence tractability properties.

It is natural to expect that one can use an analogous approach for polynomial lattice rules leading to similar results.

2 The fast reduced matrix product computation

2.1 The basic algorithm

We first present some observations which lead us to an efficient algorithm for computing XA.

Let $X = [\boldsymbol{x}_0^\top, \boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_{N-1}^\top]^\top$ be the $N \times s$-matrix whose k-th row is the k-th point of the reduced lattice point set (written as a row vector). Let $\boldsymbol{\xi}_j$ denote the j-th column of X, i.e. $X = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_s]$. Let $A = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_s]^\top$, where $\boldsymbol{a}_j \in \mathbb{R}^{1\times\tau}$ is the j-th row of A. Then we have

$$XA = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_s] \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_s \end{pmatrix} = \boldsymbol{\xi}_1\boldsymbol{a}_1 + \boldsymbol{\xi}_2\boldsymbol{a}_2 + \cdots + \boldsymbol{\xi}_s\boldsymbol{a}_s. \tag{10}$$

In order to illustrate the inherent repetitiveness of a reduced lattice point set, consider a reduction index $0 < w_j < m$ and the corresponding component z_j of the generating vector. The j-th component of the $N = b^m$ points of the reduced lattice point set (i.e., the j-th column $\boldsymbol{\xi}_j$ of X) is then given by

$$\begin{aligned} \boldsymbol{\xi}_j &:= \left(\frac{0 \cdot z_j \bmod b^{m-w_j}}{b^{m-w_j}}, \frac{1 \cdot z_j \bmod b^{m-w_j}}{b^{m-w_j}}, \ldots, \frac{(b^m-1) \cdot z_j \bmod b^{m-w_j}}{b^{m-w_j}} \right)^\top \\ &= \underbrace{(X_j, \ldots, X_j)^\top}_{b^{w_j} \text{ times}}, \end{aligned}$$

where

$$X_j = \left(0, \frac{z_j \bmod b^{m-w_j}}{b^{m-w_j}}, \ldots, \frac{(b^{m-w_j}-1) z_j \bmod b^{m-w_j}}{b^{m-w_j}} \right)^\top.$$

We will exploit this repetitive structure within the reduced lattice points to derive a fast matrix-vector multiplication algorithm.

Based on the above observations, it is possible to formulate the following algorithm to compute (10) in an efficient way. Note that for $j > s^*$ the j-th column of X consists only of zeros, so there is nothing to compute for the entries of X corresponding to these columns.

Algorithm 1 Fast reduced matrix product

Input: Matrix $A \in \mathbb{R}^{s\times\tau}$, integer $m \in \mathbb{N}$, prime b, reduction indices $0 = w_1 \le w_2 \le \cdots \le w_s$, corresponding generating vector of reduced lattice rule, $\boldsymbol{z} = (z_1, b^{w_2} z_2, \ldots, b^{w_s} z_s)$.

Set $N = b^m$ and set $P_{s^*+1} = \mathbf{0}_{1\times\tau} \in \mathbb{R}^{1\times\tau}$.
for $j = s^*$ **to** 1 **do**

- Compute the b^{m-w_j} reduced lattice points

$$X_j = \left(0, \frac{z_j \bmod b^{m-w_j}}{b^{m-w_j}}, \ldots, \frac{(b^{m-w_j}-1) z_j \bmod b^{m-w_j}}{b^{m-w_j}}\right)^\top \in \mathbb{R}^{b^{m-w_j}\times 1}.$$

- Compute P_j as

$$P_j = \overbrace{\begin{Bmatrix} \begin{pmatrix} P_{j+1} \\ P_{j+1} \\ \vdots \\ P_{j+1} \end{pmatrix} \end{Bmatrix}}^{b^{\min(w_{j+1},m)-w_j} \text{ times}} + X_j \boldsymbol{a}_j \in \mathbb{R}^{b^{m-w_j}\times\tau},$$

  where $\boldsymbol{a}_j \in \mathbb{R}^{1\times\tau}$ denotes the j-th row of the matrix A.

end for
Set $P = P_1$.

Return: Matrix product $P = XA$.

The following theorem gives an estimate of the computational cost of Algorithm 1, which shows that by using a reduced point set we can obtain an improved computation time over that in [4], which only depends on the index s^*, but not on s anymore.

Theorem 2. *Let a matrix $A \in \mathbb{R}^{s\times\tau}$, an integer $m \in \mathbb{N}$, a prime b, and reduction indices $0 = w_1 \le w_2 \le \cdots \le w_s$ be given. Furthermore, let $\boldsymbol{z} = (z_1, b^{w_2} z_2, \ldots, b^{w_s} z_s)$ be the generating vector of a reduced lattice rule corresponding to $N = b^m$ and the given reduction indices $(w_j)_{j=1}^s$. Then the matrix product $P = XA$ can be computed via Algorithm 1 using*

$$\mathcal{O}\left(\tau N \sum_{j=1}^{s^*} b^{-w_j}\right)$$

operations and requiring $\mathcal{O}(N\tau)$ storage. Here, X is the $N \times s$-matrix whose rows are the N reduced lattice points.

Proof. In the j-th step the generation of the b^{m-w_j} lattice points requires $\mathcal{O}(b^{m-w_j})$ operations and storage. The most costly operation in each step is the product $X_j \boldsymbol{a}_j$ which requires $\mathcal{O}(b^{m-w_j}\,\tau)$ operations, but this step only needs to be carried out for those j with $j \le s^*$. Summing over all $j = 1, \ldots, s^*$, the computational complexity amounts to

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

operations. Furthermore, storing the matrix P_j requires $\mathcal{O}(b^{m-w_j}\,\tau)$ space, which attains a maximum of $\mathcal{O}(b^m\,\tau)$ for $w_1 = 0$. Note that in an efficient implementation the matrices P_j are overwritten in each step and do not all have to be stored. □

In the next section we discuss the fast reduced QMC matrix-vector product where the number of points is a power of 2.

2.2 An optimized algorithm

Recall that, for a sequence $\boldsymbol{w}$ of reduction indices, we have $s^* = \max\{j \in \mathbb{N}\colon w_j < m\}$. Since for $j > s^*$ the j-th column of X consists only of zeros, we can restrict our considerations in this section to the product $\widetilde{X}\widetilde{A}$, where $\widetilde{X}$ is an $N \times s^*$-matrix, and $\widetilde{A}$ is an $s^* \times \tau$-matrix.

Assume that $0 = w_1 \le w_2 \le \cdots \le w_{s^*} < m$ and define, for $I \in \{0, 1, \ldots, m-1\}$, the quantity

$$\tau_I := \#\{j \in \{1, \ldots, s^*\} \mid w_j = I\},$$

which denotes the number of w_j which equal I. Obviously, we then have that $\sum_{I=0}^{m-1} \tau_I = s^*$.

Consider then the following alternative fast reduced matrix product algorithm.

Algorithm 2 Optimized fast reduced matrix product

Input: Matrix $\widetilde{A} \in \mathbb{R}^{s^* \times \tau}$, integer $m \in \mathbb{N}$, prime b, reduction indices $0 = w_1 \le w_2 \le \cdots \le w_{s^*} < m$, the corresponding generating vector of a reduced lattice rule, $\boldsymbol{z} = (z_1, b^{w_2} z_2, \ldots, b^{w_{s^*}} z_{s^*})$.

Set $N = b^m$, set $\widetilde{P}_m = \mathbf{0}_{1\times\tau} \in \mathbb{R}^{1\times\tau}$, and $w_{s^*+1} = m$.
for $I = m-1$ **to** 0 **do**

- Compute the matrix

$$\widetilde{X}_I = (W_1^I, \ldots, W_{\tau_I}^I) \in \mathbb{R}^{b^{m-I}\times\tau_I},$$

whose columns are the reduced lattice points

$$W_r^I = \left(0, \frac{z_{j_r} \bmod b^{m-I}}{b^{m-I}}, \ldots, \frac{(b^{m-I}-1) z_{j_r} \bmod b^{m-I}}{b^{m-I}}\right)^\top \in \mathbb{R}^{b^{m-I}\times 1},\ 1 \le r \le \tau_I,$$

and where the j_r, $1 \le r \le \tau_I$, are those indices for which $w_{j_r} = I$. If $\tau_I = 0$, then set $\widetilde{X}_I = \mathbf{0}_{b^{m-I}\times\tau_I}$.

- Compute $\widetilde{P}_I$ as

$$\widetilde{P}_I = \underbrace{\begin{pmatrix} \widetilde{P}_{I+1} \\ \vdots \\ \widetilde{P}_{I+1} \end{pmatrix}}_{b \text{ times}} + \widetilde{X}_I \widetilde{A}_I \in \mathbb{R}^{b^{m-I}\times\tau},$$

where $\widetilde{A}_I \in \mathbb{R}^{\tau_I\times\tau}$ denotes the rows of the matrix $\widetilde{A}$ that correspond to the j with $w_j = I$. If $\tau_I = 0$, then set $\widetilde{A}_I = \mathbf{0}_{\tau_I\times\tau} \in \mathbb{R}^{\tau_I\times\tau}$.

end for
Set $\widetilde{P} = \widetilde{P}_0$.

Return: Matrix product $\widetilde{P} = \widetilde{X}\widetilde{A}$.

The next theorem provides an estimate on the computation time of Algorithm 2, which again is independent of s.

Theorem 3. *Let a matrix $A \in \mathbb{R}^{s\times\tau}$, an integer $m \in \mathbb{N}$, a prime b, and reduction indices $0 = w_1 \le w_2 \le \cdots \le w_{s^*} < m$ be given. Furthermore, let $\boldsymbol{z} = (z_1, b^{w_2} z_2, \ldots, b^{w_s} z_s)$ be the generating vector of a reduced lattice rule corresponding to $N = b^m$ and the given reduction indices $(w_j)_{j=1}^s$. Then the matrix product $P = XA$ can be computed via Algorithm 2 using*

$$\mathcal{O}\left(\tau\, N m\right)$$

operations and requiring $\mathcal{O}(N\tau)$ storage. Here, X is the $N \times s$-matrix whose rows are the N reduced lattice points.

Proof. As outlined above, it is no relevant restriction to reduce the matrices X and A to an $N \times s^*$-matrix $\widetilde{X}$ and an $s^* \times \tau$-matrix $\widetilde{A}$, respectively, and then apply Algorithm 2.

In the I-th step of the algorithm, the generation of the $\tau_I b^{m-I}$ lattice points requires $\mathcal{O}(\tau_I b^{m-I})$ operations and storage. The most costly operation in each step is the product $\widetilde{X}_I \widetilde{A}_I$ which, via the fast QMC matrix product in [4], requires $\mathcal{O}((m-I)\, b^{m-I}\, \tau)$ operations. Summing over all $I = 0, \ldots, m-1$, the computational complexity amounts to

$$\mathcal{O}\left(\sum_{I=0}^{m-1} (m-I)\, b^{m-I}\, \tau\right) = \mathcal{O}\left(\tau\, b^m \sum_{I=0}^{m-1} \frac{m-I}{b^I}\right) = \mathcal{O}\left(\tau\, b^m \frac{b^2 m}{(b-1)^2}\right) = \mathcal{O}\left(\tau\, N\, m\right)$$

operations. Furthermore, storing the matrix $\widetilde{P}_j$ requires $\mathcal{O}(b^{m-I}\,\tau)$ space, which attains a maximum of $\mathcal{O}(b^m\,\tau)$ for $I = 0$. Note that in an efficient implementation the matrices $\widetilde{P}_j$ are overwritten in each step and do not all have to be stored. □

2.3 Transformations, shifting, and computation for transformation functions

In applications from mathematical finance or uncertainty quantification, the integral to be approximated is often not over the unit cube but over $\mathbb{R}^s$ with respect to a normal distribution. In order to be able to use lattice rules in this context, one has to apply a transformation and use randomly shifted lattice rules. In the following we show that the fast reduced QMC matrix-vector product can still be used in this context.

We have noted before that projections $P_j = \pi_j(P)$ of a reduced lattice point set P onto the j-th component possess a repetitive structure, that is,

$$P_j = \underbrace{(X_j, \ldots, X_j)^\top}_{b^{\min(w_j,m)} \text{ times}}$$

with

$$X_j = \left(0, \frac{z_j \bmod b^{m-\min(w_j,m)}}{b^{m-\min(w_j,m)}}, \ldots, \frac{(b^{m-\min(w_j,m)}-1) z_j \bmod b^{m-\min(w_j,m)}}{b^{m-\min(w_j,m)}}\right)^\top.$$

This repetitive structure is preserved when applying a mapping $\varphi : [0,1] \to \mathbb{R}$ elementwise to the projection P_j since

$$\varphi(P_j) = \underbrace{(\varphi(X_j), \ldots, \varphi(X_j))^\top}_{b^{\min(w_j,m)} \text{ times}}.$$

This approach also works for the map $\psi : [0,1] \to [0,1]$ with $\psi(x) = \{x + \Delta\}$, i.e, for shifting of the lattice points modulo one. In particular, this observation holds for componentwise maps of the form $\varphi : [0,1]^s \to \mathbb{R}$ with $\varphi(\boldsymbol{x}) = (\varphi_1(x_1), \ldots, \varphi_s(x_s))$ that are applied simultaneously to all N elements of the lattice point set. For a map of this form Algorithm 1 can be easily adapted by replacing X_j by $\varphi_j(X_j)$. If we wish to apply Algorithm 2 instead, the matrices $\widetilde{X}_I$ can be replaced by the correspondingly transformed matrices, however, the fast reduced QMC matrix vector product can only be used here if all components with indices in τ_I use the same transformation.

3 Reduced digital nets

In this section we present a reduced point construction for so-called digital (t, m, s)-nets. Typical examples are digital nets derived from Sobol', Faure, and Niederreiter sequences. In general, a (t, m, s)-net is defined as follows.

Given an integer $b \geq 2$, an elementary interval in $[0,1)^s$ is an interval of the form $\prod_{j=1}^{s} [a_j b^{-d_j}, (a_j + 1)b^{-d_j})$ where a_j, d_j are nonnegative integers with $0 \leq a_j < b^{d_j}$ for $1 \leq j \leq s$. Let t, m, with $0 \leq t \leq m$, be integers. Then a (t, m, s)-net in base b is a point set P_m in $[0,1)^s$ with b^m points such that any elementary interval in base b with volume b^{t-m} contains exactly b^t points of P_m.

Note that a low t-value of a (t, m, s)-net implies better equidistribution properties and usually also better error bounds for integration rules based on such nets. How to find nets with low t-values is an involved question, see, e.g., [7, 13]. Due to the important role of the t-value, one sometimes also considers a slightly refined notion of a (t, m, s)-net, which is then referred to as a $((t_{\mathfrak{u}})_{\mathfrak{u} \subseteq [s]}, m, s)$-net. The latter notion means that for any $\mathfrak{u} \neq \emptyset$, $\mathfrak{u} \subseteq [s]$, the projection of the net is a $(t_{\mathfrak{u}}, m, |\mathfrak{u}|)$-net.

The most common method to obtain (t, m, s)-nets are so-called digital constructions, yielding digital (t, m, s)-nets. These work as follows. Let b be a prime number and recall that $\mathbb{F}_b$ denotes the finite field with b elements. We identify this set with the integers $\{0, 1, \ldots, b-1\}$. We denote the (unique) b-adic digits of some $n \in \mathbb{N}$ by $\vec{n} \in \mathbb{F}_b^{\mathbb{N}}$, ordered from the least significant, that is $n = \vec{n} \cdot (1, b, b^2, \ldots)$. Here, the sum is always finite as there are only finitely many non-zero digits in $\vec{n}$. Thus, with a slight abuse of notation we write $\vec{n} \in \mathbb{F}_b^m$ if $n < b^m$. Analogously, we denote the b-adic digits of $y \in [0,1)$ by $\vec{y} \in \mathbb{F}_b^{\mathbb{N}}$, i.e. $y = \vec{y} \cdot (b^{-1}, b^{-2}, \ldots)$, with the additional constraint that $\vec{y}$ does not contain infinitely many consecutive entries equal to $b - 1$.

Given *generating matrices* $C^{(j)} = \left(C_{p,q}^{(j)}\right)_{p,q=1}^{m} \in \mathbb{F}_b^{m \times m}$ for $j = 1, \ldots, s$, a *digital net* is defined as $P_m(\{C^{(j)}\}_j) := \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{b^m-1}\}$, where $\boldsymbol{y}_n = (y_{1,n}, \ldots, y_{s,n}) \in [0,1)^s$,

$$\vec{y}_{j,n} := C^{(j)} \vec{n} \quad \text{and } y_{j,n} = \vec{y}_{j,n} \cdot (b^{-1}, b^{-2}, \ldots, b^{-m}). \tag{11}$$

From this definition, given reduction indices $\boldsymbol{w}$, one can construct a reduced digital net by setting the last $\min(w_j, m)$ rows of $C^{(j)}$ to $\mathbf{0}$. To be more precise,

$$\widehat{C}_{p,q}^{(j)} := \begin{cases} C_{p,q}^{(j)} & \text{if } p \in \{1, \ldots, m - \min(w_j, m)\}, \\ 0 & \text{if } p \in \{m - \min(w_j, m) + 1, \ldots, m\}, \end{cases} \tag{12}$$

and applying (11) with the latter choice $\widehat{C}^{(j)} = \left(\widehat{C}^{(j)}_{p,q}\right)_{p,q=1}^{m}$ of the generating matrices. Note that $\widehat{C}^{(j)}$ is just the zero matrix if $j > s^*$. For the reduced digital net, we then write $P_m(\{\widehat{C}^{(j)}\}_j) := \{\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{b^m-1}\}$.

The construction (12) allows us to generate a reduced digital net for any given digital net.

Algorithm 3 Computation of a reduced digital net

Input: Prime b, generating matrices $C^{(j)} \in \mathbb{F}_b^{m\times m}$, $j = 1, \ldots, s$, reduction indices $0 = w_1 \le w_2 \le \cdots \le w_s$.

Set $\boldsymbol{y}_0 = (0, \ldots, 0)$
for $j = 1, \ldots, s$ **do**
  for $n = 0, 1, \ldots, b^m - 1$ **do**
    Compute $\vec{z}_{j,n} = \widehat{C}^{(j)}\vec{n}$, with the choice $\widehat{C}^{(j)}$ from (12).
    Set $z_{j,n} = \vec{z}_{j,n} \cdot (b^{-1}, b^{-2}, \ldots)$
  end for
end for

Return: $P_m(\{\widehat{C}^{(j)}\}_j) = \{\boldsymbol{z}_n = (z_{1,n} \ldots, z_{s,n}) \in [0,1)^s \colon n = 0, \ldots, b^m - 1\}$.

The advantage of using a reduced digital net is that in component j all the values of the $z_{j,n}$ are in the set $\{0, 1/b^{m-\min(w_j,m)}, 2/b^{m-\min(w_j,m)}, \ldots, 1 - 1/b^{m-\min(w_j,m)}\}$. Since the digital net has b^m points, the values necessarily repeat $b^{\min(w_j,m)}$ times. This can be used to achieve a reduction in the computation of XA in the following way.

Algorithm 4 Fast reduced matrix product for digital nets

Input: Matrix $A \in \mathbb{R}^{s\times\tau}$ with j-th row vector $\boldsymbol{a}_j$, $j = 1, 2, \ldots, s$, integer $m \in \mathbb{N}$, prime b, reduction indices $0 = w_1 \le w_2 \le \cdots \le w_s$. Let $s^* \le s$ be the largest index such that $w_{s^*} < m$. Let $\{\boldsymbol{y}_n = (y_{n,1}, y_{n,2}, \ldots, y_{n,s})^\top \in [0,1)^s : n = 0, 1, \ldots, b^m - 1\}$ be a digital net.
Set $P_{s^*+1} = \boldsymbol{0} \in \mathbb{R}^{b^m\times\tau}$.
for $j = s^*$ **to** 1 **do**
- Compute the row vectors

$$\boldsymbol{c}_k = \frac{k}{b^{m-w_j}}\boldsymbol{a}_j, \quad k = 0, 1, \ldots, b^{m-w_j} - 1.$$

- Compute P_j as

$$P_j = P_{j+1} + \begin{pmatrix} \boldsymbol{c}_{\lfloor y_{0,j} b^{m-w_j}\rfloor} \\ \boldsymbol{c}_{\lfloor y_{1,j} b^{m-w_j}\rfloor} \\ \vdots \\ \boldsymbol{c}_{\lfloor y_{b^m-1,j} b^{m-w_j}\rfloor} \end{pmatrix} \in \mathbb{R}^{b^m\times\tau}.$$

end for
Set $P = P_1$.

Return: Matrix product $P = XA$.

Compared with computing XA directly, Algorithm 4 reduces the number of multiplications from $\mathcal{O}(\tau b^m)$ to $\mathcal{O}(\tau b^{m-w_j})$ in coordinate j and to $\mathcal{O}(\tau b^m \sum_{j=1}^{s^*} b^{-w_j})$ overall compared to $\mathcal{O}(s\tau b^m)$. The number of additions is the same in both instances.

The difference here to the approach for lattice point sets is that although component j has b^{w_j} repeated values, the repeating pattern in each component is different and so when we add up the vectors resulting from the different components, we do not have repetitions in general and so we do not get a reduced number of additions. The analogue to the method in Section 2.1 for lattice point sets applied to digital nets would be to delete columns of $C^{(j)}$ (rather than rows as we did in this section). The problem with this approach is that if we delete columns, then the (t, m, s)-net property of the digital net is not guaranteed anymore. A special construction of digital (t, m, s)-nets with additional properties would be needed in this case.

For the case of reduced lattice point sets, we can use Theorem 1 to obtain an error bound on the performance of the corresponding QMC rule when using (2) to approximate (1). For the case of reduced digital nets, there is no existing error bound analogous to Theorem 1. We outline the error analysis in the subsequent section.

3.1 Error analysis

Consider the case of digital nets from Algorithm 3. For this we fix $m \in \mathbb{N}$. The *weighted star discrepancy* is a measure of the worst-case quadrature error for a node set P_m, with b^m nodes, defined as

$$D^*_{b^m,\gamma}(P_m) := \sup_{x\in(0,1]^s} \max_{\emptyset\neq\mathfrak{u}\subseteq[s]} \gamma_{\mathfrak{u}} \left|\Delta_{P_m,\mathfrak{u}}(\boldsymbol{x})\right|, \tag{13}$$

where

$$\Delta_{P_m,\mathfrak{u}}(\boldsymbol{x}) := \frac{\#\{(y_1,\dots,y_s)\in P_m \colon y_j < x_j,\, \forall j\in\mathfrak{u}\}}{b^m} - \prod_{j\in\mathfrak{u}} x_j. \tag{14}$$

We additionally write $\Delta_{P_m}(\boldsymbol{x}) = \Delta_{P_m,[s]}(\boldsymbol{x})$. For all $k \in \{0,\dots,b^m-1\}$, define $\vec{k} = (\vec{\kappa}_0,\dots,\vec{\kappa}_{m-1}) \in \mathbb{F}_b^m$ its vector of b-adic digits, ordered from the least significant to the most significant. Moreover, define

$$\rho(k) = \begin{cases} 1, & \text{if } k = 0,\\ \frac{1}{b^r \sin(\pi\kappa_{r-1}/b)}, & \text{if } k = \kappa_0 + \kappa_1 b + \cdots + \kappa_{r-1}b^{r-1},\\ & \text{with } \kappa_0,\dots,\kappa_{r-2}\in\{0,1,\dots,b-1\}, \kappa_{r-1}\in\{1,\dots,b-1\}. \end{cases}$$

In the following proposition, we prove a bound on $\Delta_{P_m,\mathfrak{u}}(\boldsymbol{x})$.

Proposition 1. *Let $\widehat{P}_m := P_m(\{\widehat{C}^{(j)}\}_j) = \{\boldsymbol{z}_0,\dots,\boldsymbol{z}_{b^m-1}\}$ be generated by Algorithm 3, and let $\boldsymbol{x}\in(0,1]^s$. Let $0 = w_1 \le w_2 \le \cdots \le w_s$ and let $s^*\in[s]$ be the largest index such that $w_{s^*} < m$. Then for any $\mathfrak{u}\subseteq[s]$ with $\mathfrak{u}\neq\emptyset$ we have*

$$\left|\Delta_{\widehat{P}_m,\mathfrak{u}}(\boldsymbol{x})\right| \le \begin{cases} 1 & \text{if } \mathfrak{u}\not\subseteq[s^*],\\ 1 - \prod_{j\in\mathfrak{u}}\left(1-\frac{1}{b^{m-w_j}}\right) + \sum_{\substack{\boldsymbol{k}_{\mathfrak{u}}\in\mathbb{N}_0^{|\mathfrak{u}|}\setminus\{\boldsymbol{0}\}\\ k_j\in\{0,\dots,b^{m-w_j}-1\}\\ \sum_{j\in\mathfrak{u}}(\widehat{C}^{(j)})^\top\vec{k}_j\equiv\vec{0} \pmod b}} \prod_{j\in\mathfrak{u}}\rho(k_j), & \text{if } \mathfrak{u}\subseteq[s^*], \end{cases}$$

where $\widehat{C}^{(j)}$ *is defined in* (12).

To prove Proposition 1, we need the next elementary lemma, extending [7, Lemma 3.18], which can be verified by induction on s.

Lemma 1. *Let* J *be a finite index set and assume* $u_j, v_j \in [0,1]$, $|u_j - v_j| \le \delta_j \in [0,1]$ *for all* $j \in J$. *Then*

$$\left| \prod_{j\in J} u_j - \prod_{j\in J} v_j \right| \le 1 - \prod_{j\in J}(1-\delta_j) \le \sum_{j\in J} \delta_j.$$

Proof of Proposition 1. The bound for the case when $\mathfrak{u} \not\subseteq [s^*]$ is trivial. Hence we can now focus on the case when $\mathfrak{u} \subseteq [s^*]$. We operate along the lines of the proof of [7, Theorem 3.28]. We define the mapping $T : [0,1]^{|\mathfrak{u}|} \to [0,1]^{|\mathfrak{u}|}$ given by

$$T(\boldsymbol{x}_\mathfrak{u}) = T((x_j)_{j\in\mathfrak{u}}) = ((T_{m-w_j}(x_j))_{j\in\mathfrak{u}}),$$

where $T_v(x) = \lceil xb^v \rceil b^{-v}$.

Now, let us assume that $\boldsymbol{x}_\mathfrak{u} = (x_j)_{j\in\mathfrak{u}} \in (0,1]^{|\mathfrak{u}|}$ has been chosen arbitrarily but fixed. For short, we write $\overline{\boldsymbol{x}}_\mathfrak{u} = (\overline{x}_j)_{j\in\mathfrak{u}} := T(\boldsymbol{x}_\mathfrak{u})$.

Recall that, by the definition of the matrices $\widehat{C}^{(j)}$, $j \in \{1,\dots,s\}$, the points $\boldsymbol{z}_n = (z_{1,n},\dots,z_{s,n})$ are such that the $z_{j,n}$ have at most $m - \min(w_j, m)$ non-zero digits.

Using the triangle inequality we get

$$\left|\Delta_{\widehat{P}_m,\mathfrak{u}}(\boldsymbol{x})\right| \le \left|\Delta_{\widehat{P}_m,\mathfrak{u}}(\boldsymbol{x}) - \Delta_{\widehat{P}_m,\mathfrak{u}}(\overline{\boldsymbol{x}})\right| + \left|\Delta_{\widehat{P}_m,\mathfrak{u}}(\overline{\boldsymbol{x}})\right|.$$

For any $\boldsymbol{z}_n \in \widehat{P}_m$, we denote the b-adic digits of the j-th component $z_{j,n}$ by $z_{j,n,i}$, $i \in \{1,\dots,m\}$. By construction, we see that $z_{j,n,i} = 0$ for $i > m - \min\{w_j, m\}$. Hence

$$\widehat{P}_m \subseteq \left\{ \left(h_1 b^{-(m-\min\{w_1,m\})}, \dots, h_s b^{-(m-\min\{w_s,m\})} \right) : h_j \in \mathbb{N}_0 \right\}.$$

This implies, as $\overline{\boldsymbol{x}}_\mathfrak{u} = T(\boldsymbol{x}_\mathfrak{u})$,

$$\#\{(z_1,\dots,z_s) \in \widehat{P}_m \colon z_j < x_j,\ \forall j \in \mathfrak{u}\} = \#\{(z_1,\dots,z_s) \in \widehat{P}_m \colon z_j < \overline{x}_j,\ \forall j \in \mathfrak{u}\},$$

and thus

$$\left|\Delta_{\widehat{P}_m,\mathfrak{u}}(\boldsymbol{x}) - \Delta_{\widehat{P}_m,\mathfrak{u}}(\overline{\boldsymbol{x}})\right| = \left| \prod_{j\in\mathfrak{u}} x_j - \prod_{j\in\mathfrak{u}} \overline{x}_j \right| = \prod_{j\in\mathfrak{u}} \overline{x}_j - \prod_{j\in\mathfrak{u}} x_j \le 1 - \prod_{j\in\mathfrak{u}} \left(1 - \frac{1}{b^{m-w_j}}\right),$$

where we used Lemma 1 for the last inequality.

Since $[\boldsymbol{0}, \overline{\boldsymbol{x}})$ is a disjoint union of intervals of the form

$$J = \prod_{j=1}^{s} [\frac{h_j}{b^{m-\min(w_j,m)}}, \frac{h_j+1}{b^{m-\min(w_j,m)}}),$$

an application of [7, Lemma 3.9] implies that $\widehat{\chi_{[\boldsymbol{0},\overline{\boldsymbol{x}})}}(\boldsymbol{k}) = 0$ for all $\boldsymbol{k} \in \mathbb{N}_0^s \setminus \{\boldsymbol{0}\}$ such that $k_j \ge b^{m-\min(w_j,m)}$ for at least one j. Here $\chi_{[\boldsymbol{0},\overline{\boldsymbol{x}})}$ denotes the indicator function and $\widehat{\chi_{[\boldsymbol{0},\overline{\boldsymbol{x}})}}(\boldsymbol{k})$ are the corresponding Walsh coefficients (we use similar notation as in [7, Chapter 2]). The

complete analogue of this observation holds if we consider the projection of J, given by $J_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}}[\frac{h_j}{b^{m-\min(w_j,m)}}, \frac{h_j+1}{b^{m-\min(w_j,m)}})$, the projections $\overline{\boldsymbol{x}}_{\mathfrak{u}}$ and $\boldsymbol{k}_{\mathfrak{u}}$ of $\overline{\boldsymbol{x}}$ and $\boldsymbol{k}$, respectively, and the projections $\boldsymbol{z}_{n,\mathfrak{u}}$ of the points $\boldsymbol{z}_n$ in $\widehat{P}_m$. Then, [7, Lemmas 3.29 and 4.75] yield

$$\begin{aligned}
\left|\Delta_{\widehat{P}_m,\mathfrak{u}}(\overline{\boldsymbol{x}})\right| &= \left|\frac{1}{b^m} \sum_{\substack{\boldsymbol{k}_{\mathfrak{u}}\in\mathbb{N}_0^{|\mathfrak{u}|}\setminus\{\boldsymbol{0}\}\\ k_j\in\{0,\dots,b^{m-w_j}-1\}}} \widehat{\chi_{[\boldsymbol{0},\overline{\boldsymbol{x}}_{\mathfrak{u}})}}(\boldsymbol{k}_{\mathfrak{u}}) \sum_{n=0}^{b^m-1} \mathrm{wal}_{\boldsymbol{k}_{\mathfrak{u}}}(\boldsymbol{z}_{n,\mathfrak{u}})\right| \\
&\le \sum_{\substack{\boldsymbol{k}_{\mathfrak{u}}\in\mathbb{N}_0^{|\mathfrak{u}|}\setminus\{\boldsymbol{0}\}\\ k_j\in\{0,\dots,b^{m-w_j}-1\}}} \prod_{j\in\mathfrak{u}} \rho(k_j) \left|\frac{1}{b^m} \sum_{n=0}^{b^m-1} \mathrm{wal}_{\boldsymbol{k}}(\boldsymbol{z}_{n,\mathfrak{u}})\right| \\
&= \sum_{\substack{\boldsymbol{k}_{\mathfrak{u}}\in\mathbb{N}_0^{|\mathfrak{u}|}\setminus\{\boldsymbol{0}\}\\ k_j\in\{0,\dots,b^{m-w_j}-1\}\\ \sum_{j\in\mathfrak{u}}(\widehat{C}^{(j)})^\top\vec{k}_j\equiv\vec{0}\bmod b}} \prod_{j\in\mathfrak{u}} \rho(k_j).
\end{aligned}$$

This completes the proof. □

Let $\boldsymbol{w} = (w_j)_{j=1}^s$, $0 = w_1 \le w_2 \le \cdots \le w_s$, and let $\widehat{P}_m := P_m(\{\widehat{C}^{(j)}\}_j) = \{\boldsymbol{z}_0,\dots,\boldsymbol{z}_{b^m-1}\}$ be generated by Algorithm 3. Let $\mathfrak{u}\neq\emptyset$, $\mathfrak{u}\subseteq[s]$ be given. We then define the reduced dual net,

$$\begin{aligned}
&P^{\perp}_{m,\mathfrak{u},\boldsymbol{w}}(\{\widehat{C}^{(j)}\}_j) \\
&\qquad = \{\boldsymbol{k}_{\mathfrak{u}}\in\mathbb{N}_0^{|\mathfrak{u}|} : k_j\in\{0,\dots,b^{m-\min(w_j,m)}-1\}\,\forall j\in\mathfrak{u}, \sum_{j\in\mathfrak{u}}(\widehat{C}^{(j)})^\top\vec{k}_j\equiv\vec{0}\bmod b\},
\end{aligned}$$

and we also define the reduced dual net without zero components,

$$\begin{aligned}
&P^{\perp,*}_{m,\mathfrak{u},\boldsymbol{w}}(\{\widehat{C}^{(j)}\}_j) \\
&\qquad = \{\boldsymbol{k}_{\mathfrak{u}}\in\mathbb{N}^{|\mathfrak{u}|} : k_j\in\{1,\dots,b^{m-\min(w_j,m)}-1\}\,\forall j\in\mathfrak{u}, \sum_{j\in\mathfrak{u}}(\widehat{C}^{(j)})^\top\vec{k}_j\equiv\vec{0}\bmod b\}.
\end{aligned}$$

Furthermore, we let

$$R_{\boldsymbol{w}}(\{\widehat{C}^{(j)}\}_{j\in\mathfrak{u}}) := \sum_{\boldsymbol{k}_{\mathfrak{u}}\in P^{\perp}_{m,\mathfrak{u},\boldsymbol{w}}(\{\widehat{C}^{(j)}\}_j)\setminus\{\boldsymbol{0}\}} \prod_{j\in\mathfrak{u}} \rho(k_j). \tag{15}$$

Applying Proposition 1 to all projections of $\widehat{P}_m$ onto the sets $\mathfrak{u}\subseteq[s]$, $\mathfrak{u}\neq\emptyset$, gives the following bound on the weighted discrepancy.

Proposition 2. *Let $m\in\mathbb{N}$, let $\boldsymbol{w}$ be a given set of reduction indices with $0 = w_1 \le w_2 \le \cdots \le w_s$, let $s^*\in[s]$ be the largest index such that $w_{s^*} < m$, and let $\widehat{P}_m := P_m(\{\widehat{C}^{(j)}\}_j)$ be generated by Algorithm 3. Then,*

$$D^*_{b^m,\boldsymbol{\gamma}}(\widehat{P}_m) \le \max_{\emptyset\neq\mathfrak{u}\subseteq[s]} \gamma_{\mathfrak{u}} \begin{cases} 1 & \text{if } \mathfrak{u}\not\subseteq[s^*], \\ \left[1-\prod_{j\in\mathfrak{u}}\left(1-\frac{1}{b^{m-w_j}}\right) + R_{\boldsymbol{w}}(\{\widehat{C}^{(j)}\}_{j\in\mathfrak{u}})\right] & \text{if } \mathfrak{u}\subseteq[s^*]. \end{cases} \tag{16}$$

We will now analyze the expressions occurring in the square brackets in (16) in greater detail. To this end, we restrict ourselves to product weights in the following, i.e., we assume weights $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}} \gamma_j$ with $\gamma_1 \geq \gamma_2 \geq \cdots > 0$.

Then, using the second inequality of Lemma 1 yields for the first term for the case $\mathfrak{u} \subseteq [s^*]$ in (16),

$$\gamma_{\mathfrak{u}} \left(1 - \prod_{j\in\mathfrak{u}} \left(1 - \frac{1}{b^{m-w_j}}\right)\right) \leq \frac{1}{b^m} \gamma_{\mathfrak{u}} \sum_{j\in\mathfrak{u}} b^{w_j} \leq \frac{1}{b^m} \prod_{j\in\mathfrak{u}} \gamma_j (1 + b^{w_j}). \tag{17}$$

For the case $\mathfrak{u} \not\subseteq [s^*]$ in (16), we use that $w_j \geq m$ if $j \in \mathfrak{u} \setminus [s^*]$, and obtain for $\mathfrak{v} = \mathfrak{u} \cap [s^*]$ that

$$\gamma_{\mathfrak{u}} \leq \gamma_{\mathfrak{v}} \gamma_{\mathfrak{u}\setminus\mathfrak{v}} \frac{1}{b^m} \prod_{j\in\mathfrak{u}\setminus\mathfrak{v}} (1 + b^{w_j}) \leq \frac{1}{b^m} \prod_{j\in\mathfrak{u}} \gamma_j (1 + b^{w_j}). \tag{18}$$

Regarding the remaining term in (16), we show the following lemma.

Lemma 2. *Let $\boldsymbol{w}$ be a given set of reduction indices with $0 = w_1 \leq w_2 \leq \cdots \leq w_s$, and let $\widehat{P}_m := P_m(\{\widehat{C}^{(j)}\}_j)$ be generated by Algorithm 3. Assume that the matrices $\widehat{C}^{(1)}, \ldots, \widehat{C}^{(s)} \in \mathbb{F}_b^{m\times m}$ are the generating matrices of a digital $((t_{\mathfrak{u}})_{\mathfrak{u}\subseteq[s]}, m, s)$-net. As above, let $s^* \in [s]$ be the largest number such that $w_{s^*} < m$, and assume that $\mathfrak{v} \neq \emptyset$, $\mathfrak{v} \subseteq [s^*]$. Then,*

$$\begin{aligned} &R_{\boldsymbol{w}}(\{\widehat{C}^{(j)}\}_{j\in\mathfrak{v}}) \\ &\quad\leq \sum_{\emptyset\neq\mathfrak{p}\subseteq\mathfrak{v}} \frac{b^{t_{\mathfrak{p}}}}{b^m} \left[\frac{1}{b}\left(\frac{b^2+b}{3}\right)^{|\mathfrak{p}|} \max\left(\frac{(m-t_{\mathfrak{p}})^{|\mathfrak{p}|-1}}{(|\mathfrak{p}|-1)!}, \frac{1}{b}\right) + \left(\frac{b^2-1}{3b}\right)^{|\mathfrak{p}|} \prod_{j\in\mathfrak{p}} (m - w_j)\right]. \end{aligned} \tag{19}$$

Proof. Recall that for each $\widehat{C}^{(j)}$, $j \in \mathfrak{v}$, only the first $m - w_j$ rows of $\widehat{C}^{(j)}$ are non-zero. Consequently,

$$\begin{aligned} R_{\boldsymbol{w}}(\{C^{(j)}\}_{j\in\mathfrak{v}}) &= \sum_{\boldsymbol{k}_{\mathfrak{v}}\in P^{\perp}_{m,\mathfrak{v},\boldsymbol{w}}(\{\widehat{C}^{(j)}\})\setminus\{\boldsymbol{0}\}} \prod_{j\in\mathfrak{v}} \rho(k_j) \\ &\leq \sum_{\boldsymbol{k}_{\mathfrak{v}}\in P^{\perp}_{m,\mathfrak{v},\boldsymbol{0}}(\{\widehat{C}^{(j)}\})\setminus\{\boldsymbol{0}\}} \prod_{j\in\mathfrak{v}} \rho(k_j) \\ &= \sum_{\emptyset\neq\mathfrak{p}\subseteq\mathfrak{v}} \sum_{\boldsymbol{k}_{\mathfrak{p}}\in P^{\perp,*}_{m,\mathfrak{p},\boldsymbol{0}}(\{\widehat{C}^{(j)}\})} \prod_{j\in\mathfrak{p}} \rho(k_j). \end{aligned} \tag{20}$$

We use the estimate $(\sin(x))^{-1} \leq (\sin(x))^{-2}$ for $0 < x < \pi$ to estimate $\rho(k_j) \leq \frac{1}{b^r \sin^2(\pi\kappa_{j,a_j-1}/b)}$ for positive k_j, where we write $k_j = \kappa_{j,0} + \kappa_{j,1} b + \cdots + \kappa_{j,a_j-1} b^{a_j-1}$, with $\kappa_{j,a_j-1} \neq 0$. Now we can adapt the proof of [7, Lemma 16.40], where we replace $\frac{1}{b^{2r}}\left(\frac{1}{\sin^2(\pi\kappa_{j,a_j-1}/b)} - \frac{1}{3}\right)$ with $\frac{1}{b^r \sin^2(\pi\kappa_{j,a_j-1}/b)}$, and $[s]$ by $\mathfrak{p}$ to get the result.

To simplify the notation we prove an upper bound on the inner sum in (20) for the special case $\mathfrak{p} = [s^*]$ and assume that the underlying point set generated by $\{\widehat{C}_j\}_{j\in\mathfrak{p}} = \{\widehat{C}_j\}_{j\in[s^*]}$ is a digital (t, m, s^*)-net. Then we obtain

$$\begin{aligned}
&\sum_{\boldsymbol{k}_{\mathfrak{p}}\in P^{\perp,*}_{m,\mathfrak{p},\mathbf{0}}(\{\widehat{C}^{(j)}\})} \prod_{j\in\mathfrak{p}} \rho(k_j) \\
&= \sum_{\boldsymbol{k}_{[s^*]}\in P^{\perp,*}_{m,[s^*],\mathbf{0}}(\{\widehat{C}^{(j)}\})} \prod_{j=1}^{s^*} \rho(k_j) \\
&\le \sum_{a_1=1}^{m-w_1} \cdots \sum_{a_{s^*}=1}^{m-w_{s^*}} b^{-a_1-\cdots-a_{s^*}} \underbrace{\sum_{k_1=b^{a_1-1}}^{b^{a_1}-1} \cdots \sum_{k_{s^*}=b^{a_{s^*}-1}}^{b^{a_{s^*}}-1}}_{(\widehat{C}^{(1)})^\top \vec{k}_1+\cdots+(\widehat{C}^{(s^*)})^\top \vec{k}_{s^*}\equiv\vec{0} \pmod b} \prod_{j=1}^{s^*} \frac{1}{\sin^2(\pi\kappa_{j,a_j-1}/b)} \\
&\le \sum_{a_1=1}^{m-w_1} \cdots \sum_{a_{s^*}=1}^{m-w_{s^*}} b^{-a_1-\cdots-a_{s^*}} \left(\sum_{\kappa=1}^{b-1} \frac{1}{\sin^2(\pi\kappa/b)}\right)^{s^*} \times \\
&\quad \begin{cases} 0 & \text{if } a_1+\cdots+a_{s^*} \le m-t, \\ 1 & \text{if } m-t < a_1+\cdots+a_{s^*} \le m-t+s^*, \\ b^{a_1+\cdots+a_{s^*}-s^*-m+t} & \text{if } a_1+\cdots+a_{s^*} > m-t+s^*, \end{cases}
\end{aligned}$$

where the second inequality follows from estimating the number of solutions of the linear system $(\widehat{C}^{(1)})^\top \vec{k}_1 + \cdots + (\widehat{C}^{(s^*)})^\top \vec{k}_{s^*} \equiv \vec{0} \pmod b$, which was done in the proof of [7, Lemma 16.40]. From [7, Corollary A.23] we have $\sum_{\kappa=1}^{b-1} \frac{1}{\sin^2(\pi\kappa/b)} = \frac{b^2-1}{3}$.

Set

$$\begin{aligned}
\Sigma_1 &:= \left(\frac{b^2-1}{3}\right)^{s^*} \underbrace{\sum_{a_1=1}^{m-w_1} \cdots \sum_{a_{s^*}=1}^{m-w_{s^*}}}_{m-t+1\le a_1+\cdots+a_{s^*}\le m-t+s^*} b^{-a_1-\cdots-a_{s^*}}, \\
\Sigma_2 &:= \left(\frac{b^2-1}{3}\right)^{s^*} \underbrace{\sum_{a_1=1}^{m-w_1} \cdots \sum_{a_{s^*}=1}^{m-w_{s^*}}}_{a_1+\cdots+a_{s^*}>m-t+s^*} b^{-s^*-m+t},
\end{aligned}$$

then the inner sum in (20) is bounded by $\Sigma_1 + \Sigma_2$.

If $m-t+1-s^* \ge 0$, then we have

$$\begin{aligned}
\Sigma_1 &= \left(\frac{b^2-1}{3b}\right)^{s^*} \underbrace{\sum_{b_1=0}^{m-w_1-1} \cdots \sum_{b_{s^*}=0}^{m-w_{s^*}-1}}_{m-t+1-s^*\le b_1+\cdots+b_{s^*}\le m-t} b^{-b_1-\cdots-b_{s^*}} \\
&\le \left(\frac{b^2-1}{3b}\right)^{s^*} \sum_{\ell=m-t-s^*+1}^{m-t} b^{-\ell} \binom{\ell+s^*-1}{s^*-1} \\
&\le \left(\frac{b^2-1}{3b}\right)^{s^*} \sum_{\ell=m-t-s^*+1}^{\infty} b^{-\ell} \binom{\ell+s^*-1}{s^*-1}
\end{aligned}$$

$$\leq \left(\frac{b^2-1}{3b}\right)^{s^*} \frac{1}{b^{m-t-s^*+1}} \binom{m-t}{s^*-1} \left(\frac{b}{b-1}\right)^{s^*}$$
$$= \left(\frac{b^2+b}{3}\right)^{s^*} \frac{b^t}{b^{m+1}} \frac{(m-t)^{s^*-1}}{(s^*-1)!},$$

where we used [7, Lemma 13.24] to estimate the infinite sum.

If $m-t+1-s^* < 0$, then we have

$$\Sigma_1 \leq \left(\frac{b^2-1}{3b}\right)^{s^*} \sum_{\ell=0}^{\infty} \binom{\ell+s-1}{s-1} b^{-\ell} \leq \left(\frac{b^2-1}{3b}\right)^{s^*} \left(\frac{b}{b-1}\right)^{s^*} \leq \left(\frac{b^2+b}{3}\right)^{s^*} \frac{b^t}{b^m} \frac{1}{b^2}.$$

For Σ_2 we use the estimate

$$\Sigma_2 \leq \left(\frac{b^2-1}{3b}\right)^{s^*} \frac{b^t}{b^m} \sum_{a_1=1}^{m-w_1} \cdots \sum_{a_{s^*}=1}^{m-w_{s^*}} 1 \leq \left(\frac{b^2-1}{3b}\right)^{s^*} \frac{b^t}{b^m} \prod_{j\in[s^*]} (m-w_j).$$

The argument for the case $\mathfrak{p} = [s^*]$ can be repeated analogously for all $\emptyset \neq \mathfrak{p} \subseteq \mathfrak{v}$ in (20), by adapting notation, and in particular by replacing t by $t_{\mathfrak{p}}$. This yields the result claimed in the lemma, by plugging these estimates into (20). □

Inserting the estimates in (17), (18), and (19) into (16) yields the following theorem.

Theorem 4. *Let $\boldsymbol{w}$ be a given set of reduction indices with $0 = w_1 \leq w_2 \leq \cdots \leq w_s$, and let $\widehat{P}_m := P_m(\{\widehat{C}^{(j)}\}_j)$ be generated by Algorithm 3. Furthermore, assume product weights $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}} \gamma_j$ with $\gamma_1 \geq \gamma_2 \geq \cdots > 0$. Then,*

$$\begin{aligned} D^*_{b^m,\gamma}(\widehat{P}_m) \quad \leq \quad & \max_{\emptyset\neq\mathfrak{u}\subseteq[s]} \left[\frac{1}{b^m} \prod_{j\in\mathfrak{u}} \gamma_j (1+b^{w_j})\right] \\ & + \max_{\emptyset\neq\mathfrak{v}\subseteq[s^*]} \left[\gamma_{\mathfrak{v}} \sum_{\emptyset\neq\mathfrak{p}\subseteq\mathfrak{v}} \frac{b^{t_{\mathfrak{p}}}}{b^m} \left[\frac{1}{b}\left(\frac{b^2+b}{3}\right)^{|\mathfrak{p}|} \max\left(\frac{(m-t_{\mathfrak{p}})^{|\mathfrak{p}|-1}}{(|\mathfrak{p}|-1)!}, \frac{1}{b}\right)\right.\right. \\ & \left.\left. + \left(\frac{b^2-1}{3b}\right)^{|\mathfrak{p}|} \prod_{j\in\mathfrak{p}} (m-w_j)\right]\right]. \end{aligned} \tag{21}$$

We impose that the term

$$\max_{\emptyset\neq\mathfrak{u}\subseteq[s]} \left[\frac{1}{b^m} \prod_{j\in\mathfrak{u}} \gamma_j (1+b^{w_j})\right]$$

in (21) be bounded by κ/b^m for some constant $\kappa > 0$ independent of s. Let $j_0 \in \mathbb{N}$ be minimal such that $\gamma_j \leq 1$ for all $j > j_0$. Then we impose $\prod_{j=1}^{s} \gamma_j(1+b^{w_j}) \leq \gamma_1^{j_0} \prod_{j=1}^{s}(1+\gamma_j b^{w_j}) \leq \kappa$. Hence it is sufficient to choose $\kappa > \gamma_1^{j_0}$ and for all $j \in [s]$,

$$w_j := \min\left(\left\lfloor \log_b\left(\frac{\left(\frac{\kappa}{\gamma_1^{j_0}}\right)^{1/s} - 1}{\gamma_j}\right)\right\rfloor, m\right). \tag{22}$$

Corollary 1. *Let $\boldsymbol{\gamma}$ be product weights of the form $\gamma_{\mathfrak{u}} = \prod_{j\in\mathfrak{u}} \gamma_j$ with $\gamma_1 \geq \gamma_2 \geq \cdots > 0$ such that $\sum_{j=1}^{\infty} \gamma_j < \infty$. Let $C^{(1)}, \ldots, C^{(s)} \in \mathbb{F}_b^{m\times m}$ be the generating matrices of a digital $((t_{\mathfrak{u}})_{\mathfrak{u}\subseteq[s]}, m, s)$-net. Let the reduction indices $\boldsymbol{w}$ be chosen according to (22) and let $s^* \in [s]$ be the largest number such that $w_{s^*} < m$. Then there is a constant $C > 0$ independent of s and m, such that*

$$\begin{aligned} D^*_{b^m,\boldsymbol{\gamma}}(P_m(\{C^{(j)}\}_j)) &\leq \frac{C}{b^m} \\ &+ \max_{\emptyset\neq\mathfrak{v}\subseteq[s^*]} \left[\gamma_{\mathfrak{v}} \sum_{\emptyset\neq\mathfrak{p}\subseteq\mathfrak{v}} \frac{b^{t_{\mathfrak{p}}}}{b^m} \left[\frac{1}{b} \left(\frac{b^2+b}{3} \right)^{|\mathfrak{p}|} \max\left(\frac{(m-t)^{|\mathfrak{p}|-1}}{(|\mathfrak{p}|-1)!}, \frac{1}{b} \right) \right.\right. \\ &\left.\left. + \left(\frac{b^2-1}{3b} \right)^{|\mathfrak{p}|} \prod_{j\in\mathfrak{p}} (m - w_j) \right] \right]. \end{aligned}$$

Remark 1. Note that the choice of the quantities w_j in (22) depends on s. For sufficiently fast decaying weights γ_j, it is possible to choose the w_j such that they do no longer depend on s. Indeed, suppose, e.g., that $\gamma_j = j^{-2}$. Then we could choose the w_j such that, for some $\tau \in (1, 2)$,

$$w_j \leq \min\left(\left\lfloor \log_b \left(j^{2-\tau} \right) \right\rfloor, m \right).$$

This then yields

$$\prod_{j=1}^{s} (1 + \gamma_j b^{w_j}) \leq \exp\left(\sum_{j=1}^{s} \log(1 + \gamma_j b^{w_j}) \right) \leq \exp\left(\sum_{j=1}^{s} \gamma_j b^{w_j} \right) \leq \exp(\zeta(\tau)),$$

where $\zeta(\cdot)$ is the Riemann zeta function. This then yields a dimension-independent bound on the term $\prod_{j=1}^{s} \gamma_j(1 + b^{w_j})$ from above.

Remark 2. The term involving the maximum in the error bound of Corollary 1 crucially depends on the weights $\boldsymbol{\gamma}$ and their interplay with the t-values of the projections of $\widehat{P}_m$. In particular, small t-values in combination with sufficiently fast decaying weights should yield tighter error bounds. However, the analysis of t-values of (t, m, s)-nets is in general non-trivial (see, e.g., [7]).

4 Reduced Monte Carlo

The idea of reduction is not limited to QMC algorithms, but can also be applied to Monte Carlo algorithms, as shall be discussed in this section.

Let $N = b^m$ for some $b, m \in \mathbb{N}$ with $b \geq 2$. Further let $0 = w_1 \leq w_2 \leq w_3 \leq \cdots \leq w_s \leq w_{s+1} = m$ be some integers. Let $N_j = Nb^{-w_j} = b^{m-w_j}$ and $N_{s+1} = 1$. In particular, $N_1 = N$. Further we define $M_j = N_j/N_{j+1} = b^{w_{j+1}-w_j}$ for $j = 1, 2, \ldots, s-1$ and $M_s = N_s = b^{m-w_s}$. Then $N_j = M_j M_{j+1} \cdots M_{s-1} M_s$ and any integer $0 \leq n < N_j$ can be represented by $n = m_s N_{s+1} + m_{s-1} N_s + \cdots + m_j N_{j+1}$ with $0 \leq m_j < M_j$ for $1 \leq j \leq s$.

For each coordinate $1 \leq j \leq s$ we generate N_j i.i.d. samples $y_{j,0}, \ldots, y_{j,N_j-1}$. Different coordinates are also assumed to be independent.

Now for $0 \leq m_j < M_j$ for $1 \leq j \leq s$ let

$$x_{j,m_sN_{s+1}+m_{s-1}N_s+\cdots+m_1N_2} = y_{j,m_sN_{s+1}+\cdots+m_jN_{j+1}}. \tag{23}$$

This means that in coordinate j we only have N_j different i.i.d. samples.

4.1 Computational cost reduction

For each $0 \leq n < N$ we need to compute

$$x_{1,n}\boldsymbol{a}_1 + x_{2,n}\boldsymbol{a}_2 + \cdots + x_{s,n}\boldsymbol{a}_s,$$

where $\boldsymbol{a}_j$ is the j-th row of A. Using (23) we can write this as

$$\sum_{j=1}^{s} y_{j,m_sN_{s+1}+\cdots+m_jN_{j+1}}\boldsymbol{a}_j,$$

which we need to compute for each $0 \leq m_j < M_j$. We can do this recursively in the following way:

- First compute: $z_{s,m_s} = y_{s,m_s}\boldsymbol{a}_s$ for $0 \leq m_s < M_s$ and store the results.
- For $j = s-1, s-2, \ldots, 1$ compute:

  $$z_{j,m_sN_{s+1}+\cdots+m_jN_{j+1}} = y_{j,m_sN_{s+1}+\cdots+m_jN_{j+1}}\boldsymbol{a}_j + z_{j+1,m_sN_{s+1}+\cdots+m_{j+1}N_{j+2}}$$

  for $m_j = 0, 1, \ldots, M_j - 1$, and store the resulting vectors.

Computing the values z_{s,m_s} costs $\mathcal{O}(\tau N_s)$ operations. Computing $z_{j,m_sN_{s+1}+\cdots+m_jN_{j+1}}$ costs $\mathcal{O}(\tau N_j)$ operations.

Computing all the values therefore costs

$$\mathcal{O}\left(\tau\left(N_s + N_{s-1} + \cdots + N_1\right)\right) = \mathcal{O}\left(\tau b^m\left(b^{-w_1} + b^{-w_2} + \cdots + b^{-w_s}\right)\right)$$

operations.

If $\sum_{j=1}^{\infty} b^{-w_j} < \infty$, then the computational cost is independent of the dimension.

4.2 Error analysis

Since the samples are i.i.d., it follows that the estimator is unbiased, that is,

$$\mathbb{E}(Q(f)) = \mathbb{E}(f).$$

For a given vector $\boldsymbol{x} = (x_1, \ldots, x_s)^\top$ and $\mathfrak{u} \subseteq [s]$ let $\boldsymbol{x}_\mathfrak{u} = (x_j)_{j\in\mathfrak{u}}$ and $\boldsymbol{x}_{-\mathfrak{u}} = (x_j)_{j\notin\mathfrak{u}}$. We now consider the variance of the estimator. Let

$$\mu_u := \mathbb{E}_{\boldsymbol{x}_\mathfrak{u}}\mathbb{E}_{\boldsymbol{x}_{-\mathfrak{u}}}\mathbb{E}_{\boldsymbol{y}_{-\mathfrak{u}}} f = \int\int f(\boldsymbol{x}_\mathfrak{u}\mathfrak{u}, \boldsymbol{x}_{-\mathfrak{u}})\,\mathrm{d}\boldsymbol{x}_{-\mathfrak{u}} \int f(\boldsymbol{x}_\mathfrak{u}, \boldsymbol{y}_{-\mathfrak{u}})\,\mathrm{d}\boldsymbol{y}_{-\mathfrak{u}}\,\mathrm{d}\boldsymbol{x}_\mathfrak{u}.$$

For instance, $\mu_\emptyset = (\mathbb{E}(f))^2$ and $\mu_{\{1,\ldots,s\}} = \int f^2$.

In classical Monte Carlo integration, one studies the variance $\mathrm{Var}(Q(f)) = (\mu_{\{1,\ldots,s\}} - \mu_\emptyset)/N$. We now show how the reduced MC construction influences the variance.

Theorem 5. *The variance of the reduced Monte Carlo estimator is given by*

$$\mathrm{Var}(Q(f)) = \sum_{k=1}^{s} \mu_{\{k,k+1,\ldots,s\}} \prod_{j=k}^{s} M_j^{-1} \left(1 - \frac{1}{M_{k-1}}\right) - \frac{\mu_\emptyset}{M_s},$$

where we set $1 - \frac{1}{M_0} = 1$.

Proof. The variance of $Q(f)$ can be written as $\mathbb{E}(Q^2(f)) - (\mathbb{E}(Q(f)))^2$. The last term $(\mathbb{E}(Q(f)))^2$ equals $(\mathbb{E}(f))^2 = \mu_\emptyset$.

We have

$$Q(f) = \frac{1}{M_1} \sum_{m_1=0}^{M_1-1} \cdots \frac{1}{M_s} \sum_{m_s=0}^{M_s-1} f(y_{1,n_1}, \ldots, y_{s,n_s}),$$

where $n_j = m_s N_{s+1} + \cdots + m_j N_{j+1}$. Hence

$$\begin{aligned} Q^2(f) =& \frac{1}{M_1^2} \sum_{m_1,m_1'=0}^{M_1-1} \cdots \frac{1}{M_s^2} \sum_{m_s,m_s'=0}^{M_s-1} f(y_{1,n_1}, \ldots, y_{s,n_s}) f(y_{1,n_1'}, \ldots, y_{s,n_s'}) \\ =& \sum_{u \subseteq \{1,\ldots,s\}} \frac{1}{N^2} \sum f(y_{1,n_1}, \ldots, y_{s,n_s}) f(y_{1,n_1'}, \ldots, y_{s,n_s'}), \end{aligned}$$

where the second sum is over all $0 \le m_j, m_j' < M_j$ such that $m_j = m_j'$ for $j \in u$ and $m_j \neq m_j'$ for $j \notin u$.

Let $1 \le k \le s$. If $m_j = m_j'$ for $k \le j \le s$, then $n_j = n_j'$ for $k \le j \le s$ and if $m_{k-1} \neq m_{k-1}'$, then $n_i \neq n_i'$ for $1 \le i < k$. In this case

$$\mathbb{E}\left(f(y_{1,n_1}, \ldots, y_{s,n_s}) f(y_{1,n_1'}, \ldots, y_{s,n_s'})\right) = \mu_{\{k,k+1,\ldots,s\}}.$$

The number of such instances is given by $\prod_{j=k}^{s} M_j (M_{k-1}^2 - M_{k-1}) \prod_{j=1}^{k-2} M_j^2$. Since $N = M_1 M_2 \cdots M_s$, we obtain $\prod_{j=k}^{s} M_j (M_{k-1}^2 - M_{k-1}) \prod_{j=1}^{k-2} M_j^2 N^{-2} = \prod_{j=k}^{s} M_j^{-1} (1 - M_{k-1}^{-1})$.

If $m_s \neq m_s'$ we obtain that

$$\mathbb{E}\left(f(y_{1,n_1}, \ldots, y_{s,n_s}) f(y_{1,n_1'}, \ldots, y_{s,n_s'})\right) = \mu_\emptyset.$$

This case occurs $(M_s^2 - M_s) \prod_{j=1}^{s-1} M_j^2$ times, and therefore $(M_s^2 - M_s) \prod_{j=1}^{s-1} M_j^2 N^{-2} = 1 - M_s^{-1}$.

Using the linearity of expectation we obtain the formula. □

5 Numerical experiments

In this section we give exemplary numerical results regarding the use of reduced rank-1 lattice point sets for matrix products, as outlined in Section 2.

5.1 Reduced matrix-vector products

In each case, we compute the generating vectors $\boldsymbol{z} = (z_1 b^{w_1}, \ldots, z_s b^{w_s})$ depending on the reduction indices w_j via a reduced CBC construction with product weights $\gamma_j = 0.7^j$, as developed in [2]. For a fair comparison, we do not include in the timings the construction of $\boldsymbol{z}$ and we average the computing times over 10 runs. Computations are run using MATLAB 2019a on an Octa-Core (Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz) laptop.

As a first example, we illustrate the benefit of Algorithm 1 compared to the standard matrix-vector product to compute $P = XA$ for $A \in \mathbb{R}^{s \times \tau}$. In Figure 1 we compare different combinations of s, m, for the choice of reduction indices $w_j = \min(\lfloor \log_2(j) \rfloor, m)$ and fixed $b = 2$. We repeat the same experiment on Algorithm 2 with the same settings.

In Figures 1–3, the blue graphs show the results for the reduced matrix-matrix product according to Algorithm 1, the red graphs show the results for the optimized reduced matrix-matrix product according to Algorithm 2, and the light brown graphs show the results for a straightforward implementation of the matrix-matrix product without any adjustments.

We conclude that the computational saving due to Algorithms 1 and 2 is more pronounced for larger m, s. Note that the right plot is in semi-logarithmic scale.

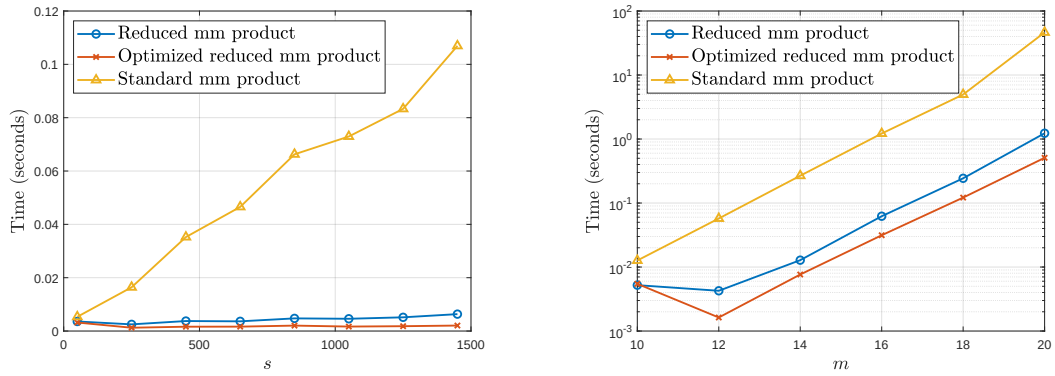

Figure 1: $m = 12$, $\tau = 20$, varying s (left) and $s = 800$, $\tau = 20$, varying m (right).

Next we study in Figure 2 the behavior as the size τ increases. Also here, we see a clear advantage of Algorithms 1 and 2 over a straightforward implementation of the matrix-matrix product.

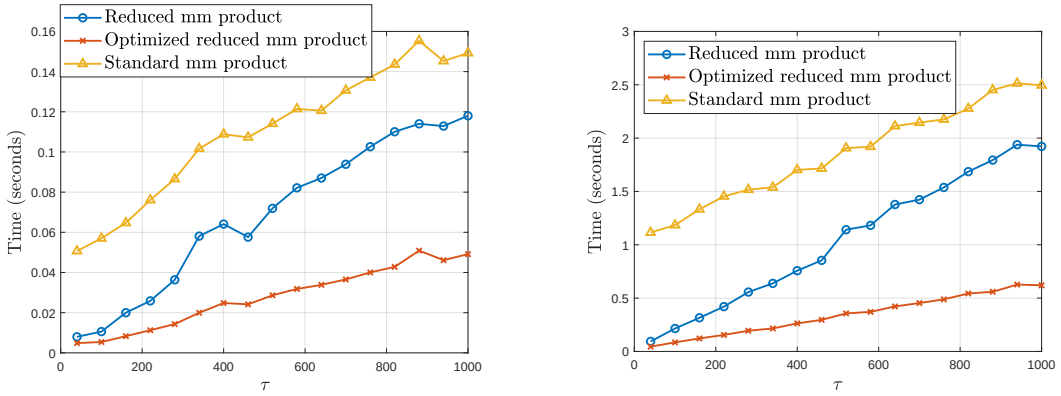

Figure 2: $s = 800$, varying τ for $m = 12$ (left) and $m = 16$ (right).

When the reduction is less aggressive, that is, w_j increases more slowly, the benefit is still considerable for large s especially for Algorithm 2, see Figure 3.

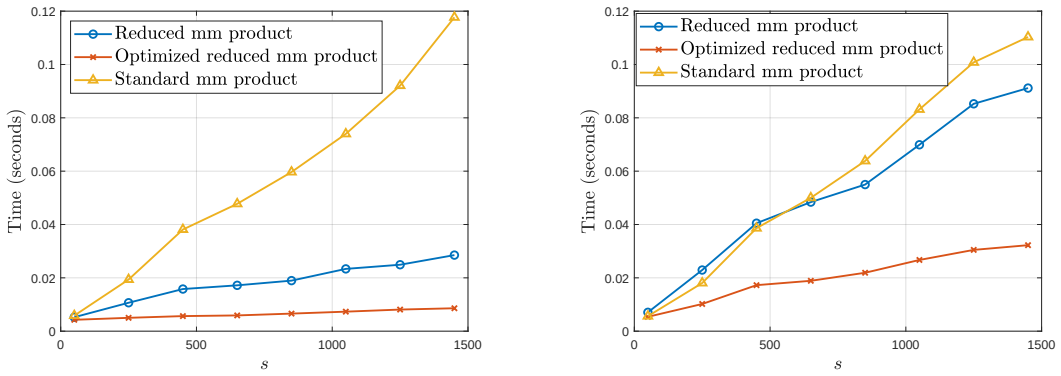

Figure 3: $m = 12$, $\tau = 20$, varying s for $w_j = \min\left(\left\lfloor \log_2(j^{1/2}) \right\rfloor, m\right)$ (left) and $w_j = \min\left(\left\lfloor \log_2(j^{1/4}) \right\rfloor, m\right)$ (right).

We now test the reduced matrix-vector product for Monte Carlo integration with respect to the normal distribution. As an example, we consider the pricing of a basket option [9, Section 3.2.3]. We define the payoff $H(S) = \max(\frac{1}{s}\sum_{j=1}^{s} S_j(T) - K, 0)$, where $S_j(T)$ is the price of the j-th asset at maturity T. Under the Black and Scholes model with zero interest rate we have $S_j(T) = S_j(0)\exp(-\Sigma_{jj}/2T + W_j\sqrt{T})$, where $W_j = LZ$, $Z \sim \mathcal{N}(0, \mathrm{Id}_{s\times s})$ and $LL^T = \Sigma$ is the covariance matrix of the random vector $S = (S_1, \ldots, S_s)$.

We set $S_j(0) = 100$ for all j, $s = 10, T = 1$, strike price $K = 110$, and as the covariance

matrix we pick $\sigma = 0.4, \rho = 0.2$ and

$$\Sigma = \begin{pmatrix} \sigma & \rho & & & \\ \rho & \sigma & \rho & & \\ & \ddots & \ddots & \ddots & \\ & & \rho & \sigma & \rho \\ & & & \rho & \sigma \end{pmatrix}.$$

We approximate the option price $\mathbb{E}(S) \approx \frac{1}{b^m} \sum_{k=1}^{b^m} S_j(0) \exp(-\sigma/2T + L\boldsymbol{x}_k\sqrt{T})$, with $\boldsymbol{x}_k$ random samples from Z. The main work is to compute $XL^\top$ (recall that X was defined in (3)) and thus the reduced matrix-vector multiplication can be beneficial in this example. Results for different choices of reduction indices w_j are displayed in Figure 4, where we plot the mean error over $R = 5$ repetitions for different values of reduction indices, using $Rb^m, m = 25$, Monte Carlo samples for the reference value. Note that the performance of QMC methods in this illustration appears to be not particularly strong as compared to standard Monte Carlo, as we consider a setting without coordinate weights, which usually is unfavorable for QMC methods.

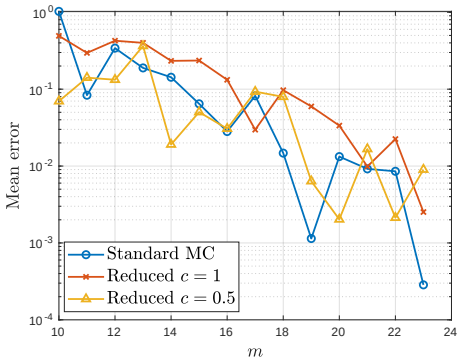

Figure 4: Option pricing example: standard Monte Carlo (corresponding to $c = 0$, compared with reduced Monte Carlo for $w_j = \min\left(\lfloor \log_2(j^c) \rfloor, m\right)$, $c \in \{1, 0.5\}$.

Acknowledgements

Josef Dick is supported by the Australian Research Council Discovery Project DP220101811. Adrian Ebert and Peter Kritzer acknowledge the support of the Austrian Science Fund (FWF) Project F5506, which is part of the Special Research Program "Quasi-Monte Carlo Methods: Theory and Applications". Furthermore, Peter Kritzer has partially been supported by the Austrian Science Fund (FWF) Project P34808. For the purpose of open access, the authors have applied a CC BY public copyright licence to any author accepted manuscript version arising from this submission.

References

[1] R. Cools, F.Y. Kuo, D. Nuyens, G. Suryanarayana. Tent-transformed lattice rules for integration and approximation of multivariate non-periodic functions. J. Complexity

36, 166–181, 2016.

[2] 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.

[3] J. Dick, P. Kritzer, F. Pillichshammer. *Lattice Rules*. Springer, Cham, 2022.

[4] J. Dick, F.Y. Kuo, Q.T. Le Gia, Ch. Schwab. Fast QMC matrix-vector multiplication. SIAM J. Sci. Comput. 37(3), A1436–A1450, 2015.

[5] J. Dick, F.Y. Kuo, I.H. Sloan. High-dimensional integration: The quasi-Monte Carlo way. Acta Numer. 22, 133–288, 2013.

[6] J. Dick, D. Nuyens, F. Pillichshammer. Lattice rules for nonperiodic smooth integrands. Numer. Math. 126, 259–291, 2014.

[7] J. Dick, F. Pillichshammer. *Digital Nets and Sequences. Discrepancy Theory and Quasi-Monte Carlo Integration.* Cambridge University Press, Cambridge, 2010.

[8] J. Dick, I.H. Sloan, X. Wang, H. Woźniakowski. Good lattice rules in weighted Korobov spaces with general weights. Numer. Math. 103, 63–97, 2006.

[9] P. Glasserman. *Monte Carlo Methods in Financial Engineering.* Applications of Mathematics, 53. Stochastic Modelling and Applied Probability. Springer-Verlag, New York, 2004.

[10] F.J. Hickernell. A generalized discrepancy and quadrature error bound. Math. Comp. 67, 299–322, 1998.

[11] F.J. Hickernell, H. Woźniakowski. Integration and approximation in arbitrary dimensions. High dimensional integration. Adv. Comput. Math. 12, 25–58, 2000.

[12] F.Y. Kuo, D. Nuyens, Application of quasi-Monte Carlo methods to elliptic PDEs with random diffusion coefficients: a survey of analysis and implementation. Found. Comput. Math. 16, no. 6, 1631–1696, 2016.

[13] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* CBMS-NSF Regional Conference Series in Applied Mathematics, 63. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1992.

[14] E. Novak, H. Woźniakowski. *Tractability of Multivariate Problems, Volume II: Standard Information for Functionals.* EMS, Zurich, 2010.

[15] I.H. Sloan, S. Joe. *Lattice methods for multiple integration.* The Clarendon Press, Oxford University Press, New York, 1994.

Authors' addresses:

Josef Dick
School of Mathematics and Statistics
University of New South Wales (UNSW)

Sydney, NSW, 2052, Australia
josef.dick@unsw.edu.au

Adrian Ebert
Johann Radon Institute for Computational and Applied Mathematics (RICAM)
Austrian Academy of Sciences
Altenbergerstr. 69, 4040 Linz, Austria
and
Centrica Business Solutions
Roderveldlaan 2, 2600 Antwerpen, Belgium adrian.ebert@hotmail.com

Lukas Herrmann
Johann Radon Institute for Computational and Applied Mathematics (RICAM)
Austrian Academy of Sciences
Altenbergerstr. 69, 4040 Linz, Austria
lukas.herrmann@alumni.ethz.ch

Peter Kritzer
Johann Radon Institute for Computational and Applied Mathematics (RICAM)
Austrian Academy of Sciences
Altenbergerstr. 69, 4040 Linz, Austria
peter.kritzer@oeaw.ac.at

Marcello Longo
Seminar for Applied Mathematics
ETH Zürich
Rämistrasse 101, 8092 Zürich, Switzerland
marcello.longo@sam.math.ethz.ch