Fast component-by-component construction for (non-)primes [⋆]

Dirk Nuyens, Ronald Cools

Dept. of Computer Science, K.U.Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium

Abstract

The component-by-component construction algorithm for rank-1 lattices can be formulated elegantly as a repeated matrix-vector product. As was shown in an earlier paper, this matrix-vector product can be done in time $O(n \log(n))$ for n prime. Here we extend this result to general n using facts from algebra to obtain a construction cost of $O(sn \log(n))$ for a rank-1 lattice in s dimensions with n points.

As was the case for n prime, the main calculation cost is significantly reduced by using fast Fourier transforms in the matrix-vector calculation. The number of fast Fourier transforms is dependent on the number of divisors of n and the number of prime factors of n. It is believed that the intrinsic structure present in rank-1 lattices and exploited by this fast construction method will deliver new insights in the applicability of these lattices.

Key words: Numerical integration, Quadrature and cubature formulas, Quasi-Monte Carlo, Rank-1 lattices, Fast component-by-component construction, Analysis of algorithms
1991 MSC: 65D30, 65D32, 68W40

[⋆] This research is part of a project financially supported by the Onderzoeksfonds K.U.Leuven / Research Fund K.U.Leuven

Email addresses: `dirk.nuyens@cs.kuleuven.ac.be` (Dirk Nuyens), `ronald.cools@cs.kuleuven.ac.be` (Ronald Cools).

Dagstuhl Seminar Proceedings 04401
Algorithms and Complexity for Continuous Problems
http://drops.dagstuhl.de/opus/volltexte/2005/148

1 Introduction

The application of this paper is the approximation of an s-dimensional integral over the unit cube by an equal weight cubature rule,

$$I(f) = \int_{[0,1)^s} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \quad \approx \quad Q(f) = \frac{1}{n} \sum_{\boldsymbol{x}_k \in P_n} f(\boldsymbol{x}_k)\,, \tag{1}$$

where the n evaluation points are a rank-1 lattice

$$P_n = \left\{ \frac{k \cdot \boldsymbol{z}}{n} : 0 \leq k < n \right\}, \tag{2}$$

and by $k \cdot \boldsymbol{z}$ we mean (componentwise) multiplication modulo n. The integer vector $\boldsymbol{z}$ is called the generating vector of the lattice and its components come from the set $U_n = \{z \in \mathbb{Z}_n : \gcd(z,n) = 1\}$. This set contains the *units* of $\mathbb{Z}_n$ and assures us that $(k \cdot z_j)/n$ is a permutation of the equispaced distribution k/n in each dimension j, where $k = 0, \ldots, n-1$. This generating vector is chosen such as to minimize a certain error measure for the cubature rule (1). Here we minimize the worst-case error for all functions in the unit ball of a Hilbert space $\mathcal{H}$:

$$e(P_n, \mathcal{H}) = \sup\{|I(f) - Q(f)| : f \in \mathcal{H}, \|f\| \leq 1\}\,.$$

We take the same scenery as in [10], that is we use a shift-invariant tensor product reproducing kernel Hilbert space, with kernel

$$K_{s,\boldsymbol{\gamma}}(\boldsymbol{x}) = \prod_{j=1}^{s} K_{1,\gamma_j}(x_j)\,, \qquad\qquad K_{1,\gamma} = 1 + \gamma\, \omega(x)\,,$$

in which the worst-case error for n sample points in s dimensions can be written as a function of $\boldsymbol{z}$ as

$$e_{n,s}(z_1, z_2, \ldots, z_s) = \left[-1 + \frac{1}{n} \sum_{k=0}^{n-1} p_{s-1}(k) \left(1 + \gamma_s\, \omega\left(\frac{k \cdot z_s}{n} \right) \right) \right]^{1/2}, \tag{3}$$

and the n-vector $\boldsymbol{p}_{s-1}$ is recursively defined as

$$p_{s-1}(k) = p_{s-2}(k) \left(1 + \gamma_{s-1}\, \omega\left(\frac{k \cdot z_{s-1}}{n} \right) \right), \qquad\qquad p_0(k) = 1\,. \tag{4}$$

The function ω in these formulas can be taken arbitrary. The most often used function spaces are the weighted Korobov space, for periodic functions, and the weighted Sobolev space, for non-periodic functions. For example, the reproducing kernel for a Korobov space with smoothness parameter $\alpha = 2$ is given by

$$K_{1,\gamma}(x) = 1 + \gamma\, 2\pi^2\, B_2(x)\,, \qquad\qquad B_2(x) = x^2 - x + 1/6\,.$$

We consider so-called weighted function spaces, where the weights denote the relative importance of certain coordinates in the function space. The weights used here are product-type weights [12] where the γ_j are taken as a decaying sequence of positive weights,

$$\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_s \geq 0\,,$$

to denote that successive coordinates are less and less important.

In Section 2 we will introduce the component-by-component algorithm in its matrix-vector form. After introducing some necessary theory in Section 3 and Section 4, we will show in Section 5 how to obtain a fast, $O(sn\log(n))$, component-by-component construction algorithm by providing a fast matrix-vector multiplication for general n. In Section 6 we will illustrate the techniques which were introduced in the previous sections for different cases. We conclude the paper in Section 7.

2 A matrix-vector form of the component-by-component algorithm

Our goal is to select a generating vector $\boldsymbol{z}$ which tries to minimize the worst-case error, defined in (3). The component-by-component algorithm (introduced in [11]) finds values for the components of the generating vector one component at a time, while keeping the previously made choices fixed. Since we start with z_1 to construct a 1-dimensional rule, then go on to find z_2 for a 2-dimensional rule and so on, it is assumed that the components are ordered by importance (cf. the weights γ_j form a decaying sequence).

So in iteration s we have to calculate the worst-case error (3) for each possible candidate $z_s \in U_n$. Abstracting out the γ_s and the summation of the constant 1 in (3) gives a formula of the form

$$v_s(z) = \sum_{k=0}^{n-1} p_{s-1}(k)\, \omega\left(\frac{k \cdot z}{n}\right)\,, \qquad\qquad \forall z \in U_n\,,$$

which can easily be identified with a matrix-vector product (for all z at once):

$$\boldsymbol{v}_s = \Omega_n \cdot \boldsymbol{p}_{s-1}\,, \qquad\qquad \Omega_n = \left[\omega\left(\frac{k \cdot z}{n}\right)\right]_{\substack{z \in U_n \\ k \in \mathbb{Z}_n}}\,.$$

We have taken the liberty to define this matrix Ω_n without specifying the iteration order of the elements $z \in U_n$ and $k \in \mathbb{Z}_n$ which define it. For now we take this order arbitrary, although for correctness the ordering of $\boldsymbol{v}_s$ and $\boldsymbol{p}_{s-1}$ must match those of U_n and $\mathbb{Z}_n$ respectively, and this is silently assumed from now on.

After this matrix-vector product we search the minimum value of $\boldsymbol{v}_s$, i.e. $v_s(z^\star)$, and choose the corresponding z candidate as the optimal choice for $z_s = z^\star$. It can be seen easily that the minimum index $z^\star$ for $\boldsymbol{v}_s$ is also the index for the worst-case error $\boldsymbol{e}_s$ with minimum $e_s(z^\star)$. Thus:

$$z_s = \operatorname*{argmin}_{z \in U_n} e_s(z) \,.$$

Once we know z_s we can overwrite the old $\boldsymbol{p}$-vector, $\boldsymbol{p}_{s-1}$, with the new $\boldsymbol{p}_s$ using (4). This is in fact (almost) the same as multiplying each element in $\boldsymbol{p}_{s-1}$ by the corresponding element of the row of Ω_n which corresponds to our choice of z_s:

$$p_s(k) = (1 + \gamma_s \, \Omega_n(z_s, k)) \cdot p_{s-1}(k) \,, \qquad \forall k \in \mathbb{Z}_n \,.$$

This can be written in matrix-vector lingo as a product with a diagonal matrix

$$\boldsymbol{p}_s = \operatorname{diag}(1 + \gamma_s \, \Omega_n(z_s, :)) \cdot \boldsymbol{p}_{s-1} \,,$$

where $\Omega_n(z_s, :)$ means the z_s-th row of the matrix Ω_n.

This brings us to a short and concise formulation in Algorithm 1 of the component-by-component algorithm.

Algorithm 1 The component-by-component algorithm

```
for $s = 1$ to $s_{\max}$ do
    $\boldsymbol{e}_s^2 = -1 + \frac{1}{n}\,(1 + \gamma_s \, \Omega_n) \cdot \boldsymbol{p}_{s-1}$
    $z_s = \operatorname{argmin} e_s^2(z)$
    $\boldsymbol{p}_s = \operatorname{diag}\,(1 + \gamma_s \, \Omega_n(z_s, :)) \cdot \boldsymbol{p}_{s-1}$
end for
```

By simple inspection of the algorithm, we find that the major cost in constructing such a rank-1 lattice rule is concentrated in the matrix-vector multiplication. A general matrix-vector product has time complexity $O(n^2)$ for a matrix of order n, and so the obvious component-by-component construction of a rank-1 lattice with n points in s dimensions is $O(sn^2)$. A more precise construction cost can be derived by using the actual size of the matrix Ω_n, which is $|U_n| \times |\mathbb{Z}_n| = \phi(n) \times n$, and thus the construction cost is $O(s\phi(n)n)$, where ϕ is the Euler totient function. This means that the construction cost is $O(sn^{2-\delta})$ with $0 < \delta \leq 1/2$ when using a general matrix-vector product.

Since our matrix Ω_n has at most n different elements, and since in such a case it is often possible to do a matrix-vector product in $O(n \log(n))$ instead of $O(n^2)$, we could of course hope that component-by-component construction could be done in $O(sn \log(n))$. Such a technique was introduced in [10] for n prime.

3 Preliminaries

Define the index-matrix Ξ_n to represent the structure of Ω_n. This index-matrix has the same size as Ω_n where at position (k, z) we do not have the value of $\omega((k \cdot z)/n)$, but just the index $i = k \cdot z \bmod n$:

$$\Xi_n = \Big[k \cdot z \bmod n\Big]_{\substack{z \in U_n \\ k \in \mathbb{Z}_n}} .$$

We can form the matrix Ω_n (assuming the same ordering of the index sets U_n and $\mathbb{Z}_n$) in a simple way from Ξ_n, by the application of the kernel function ω operating elementwise

$$\Omega_n = \omega(\Xi_n / n) .$$

There is a matrix homomorphism from the matrix Ξ_n to the matrix Ω_n, i.e. the modulo multiplication structure present in Ξ_n is preserved in Ω_n. We will formalize this matrix homomorphism in the following definition were we use the term *codomain* to denote the set of entries in the matrix and the term *domain* to denote its index set.

Definition 1 (Matrix homomorphism). *A mapping φ from the codomain of A onto the codomain of B defines a matrix homomorphism if there exist mappings σ_r and σ_c from the domain of A onto the domain of B such that*

$$\forall (i,j) \in \mathrm{domain}(A) : A(i,j) = t \Leftrightarrow B(\sigma_r(i), \sigma_c(j)) = \varphi(t) .$$

Similarly we can define a *matrix isomorphism* which states that two matrices are isomorphic when the mappings σ_r, σ_c and φ are one-to-one and onto. Note that these definitions are chosen in such a way that the Cayley tables of two isomorphic groups A and B are isomorphic matrices and vice versa.

Please note that we take a very liberate view on the matrices we are using. A more suitable way to look at these matrices might be as functions from one finite domain (the domain of the matrix) onto another finite domain (the codomain of the matrix). (A trivial extension is also possible to multi-dimensional matrices.) Thus we could define Ω_n and Ξ_n as

$$\begin{aligned} \Xi_n &: U_n \times \mathbb{Z}_n \to \mathbb{Z}_n &&: (z,k) \mapsto z \cdot k \bmod n \,, \\ \Omega_n &: U_n \times \mathbb{Z}_n \to \mathbb{R}(\Omega_n) &&: (z,k) \mapsto \omega((z \cdot k)/n) = \omega(\Xi_n(z,k)/n) \,, \end{aligned}$$

where $\mathbb{R}(\Omega_n)$ means the set $\mathbb{R}$ restricted to the actual values in the matrix Ω_n.

For completeness we will now give some basic abstract algebra results which we will use further on, and which can all be found in a standard algebra textbook, e.g. [7].

Definition 2 (Cyclic group). *A group G is called cyclic whenever all its elements can be generated by the powers of an element $g \in G$, called a* generator*, and thus $G = \langle g \rangle = \{g^k : k \in \mathbb{Z}\}$.*

Corollary 1 (Cyclicness of U_n). *The multiplicative group $U_n = \{z \in \mathbb{Z}_n : \gcd(z,n) = 1\}$, with order given by the Euler totient function as $|U_n| = \phi(n)$, is cyclic whenever*

$$n = 2, 4, p^k \text{ or } 2p^k \,,$$

with p an odd prime. A generator for the cyclic group U_n is called a primitive root modulo n.

An algorithm to find a primitive root modulo n can be found in [1]. If we have a generator g for the cyclic group U_n (with n given as in Corollary 1) then we can list the elements of U_n in natural order of this generator as

$$U_n = \langle g \rangle = \{g^0, g^1, g^2, \ldots, g^{\phi(n)-1}\} \,.$$

We will now make a connection between the Cayley table of a cyclic group and a circulant matrix. For more about circulant matrices see e.g. [2].

Definition 3 (Circulant matrix). *A circulant matrix $C_m = \operatorname{circ}(\boldsymbol{c})$ of order m is a matrix defined by the m elements in the vector $\boldsymbol{c}$ as*

$$[C_m]_{k,\ell} = c_{k-\ell \bmod m} \,.$$

In other words every diagonal of a circulant matrix consists of the same element and each column is a cyclic downshifted version from the previous column. Thus the first column is just the vector $\boldsymbol{c}$ and the last column is this vector upside down.

The Cayley table of a cyclic group can be made to look like a circulant matrix. Consider a cyclic group G with a generator g. We can then picture the Cayley table as the left part in (5).

$$\begin{array}{c|ccccc}
\cdot & g^0 & g^1 & g^2 & \cdots & g^{-1} \\ \hline
g^0 & g^0 & g^1 & g^2 & \cdots & g^{-1} \\
g^1 & g^1 & g^2 & g^3 & \iddots & g^0 \\
g^2 & g^2 & g^3 & \iddots & \iddots & g^1 \\
\vdots & \vdots & \iddots & \iddots & \iddots & \vdots \\
g^{-1} & g^{-1} & g^0 & g^1 & \cdots & g^{-2}
\end{array}
\quad \simeq \quad
\begin{array}{c|ccccc}
\cdot & g^0 & g^{-1} & g^{-2} & \cdots & g^1 \\ \hline
g^0 & g^0 & g^{-1} & g^{-2} & \ldots & g^1 \\
g^1 & g^1 & g^0 & g^{-1} & \ddots & \vdots \\
g^2 & g^2 & g^1 & \ddots & \ddots & g^{-2} \\
\vdots & \vdots & \ddots & \ddots & \ddots & g^{-1} \\
g^{-1} & g^{-1} & \ldots & g^2 & g^1 & g^0
\end{array} \tag{5}$$

By inspection we see that using the natural order of a generator results in constant anti-diagonals. By using the negative powers of the generator for the columns of the Cayley table, and keeping the positive powers of the generator for the rows, we obtain the table depicted at the right of (5). We now have constant diagonals and this table can be interpreted as a circulant matrix.

Theorem 1 (Diagonalization of a circulant matrix). *A circulant matrix has a similarity transform with the Fourier matrix as its eigenvectors*

$$C_m = F_m^{-1} \cdot D \cdot F_m \,,$$

and its eigenvalues are given by the discrete Fourier transform of its defining elements in the vector $\boldsymbol{c}$

$$D = \operatorname{diag}(F_m \cdot \boldsymbol{c}) \,.$$

Corollary 2 (Fast matrix-vector product with a circulant matrix). *A matrix-vector product with a circulant matrix C_m takes time $O(m \log(m))$ instead of $O(m^2)$ when using its similarity transform and a fast Fourier algorithm:*

$$\begin{aligned} C_m \cdot \boldsymbol{x} &= F_m^{-1} \cdot D \cdot F_m \cdot \boldsymbol{x} \\ &= \mathrm{IFFT}(\operatorname{diag}(\mathrm{FFT}(\boldsymbol{c})) \cdot \mathrm{FFT}(\boldsymbol{x})) \,. \end{aligned}$$

Note that a fast Fourier transform in time $O(m \log(m))$ is always possible when an m-point discrete Fourier transform is necessary. For such an implementation see e.g. [6].

4 Partitioning the index-matrix into circulant blocks

The technique which we will use for the fast construction will be based on block-partitioning the matrix Ξ_n into smaller matrices which are isomorphic to (block) circulant matrices (and thus isomorphic to cyclic groups). We will do this partitioning based on the structure present in Ξ_n and therefore we need not be concerned anymore with the actual matrix Ω_n. The derived techniques will work for any matrix Ω_n as long as there is a matrix homomorphism from Ξ_n to Ω_n.

For the partitioning of Ξ_n into smaller blocks we need to partition its domain, which is $U_n \times \mathbb{Z}_n$. We will handle these two sets separately. But first we will introduce two alternatives for looking at the numbers in $\mathbb{Z}_n$.

A first useful way of looking at $v \in \mathbb{Z}_n$ is given by its prime factorization

$$v = \underbrace{\prod_{p\,|\,n} p^{v_p}}_{\text{divisors of } n} \cdot \underbrace{\prod_{p\nmid n} p^{v_p}}_{\text{units of } n} , \qquad p \text{ prime, } p \leq n \text{ and each } v_p \geq 0\,.$$

Since the divisors of n are all those numbers which have the same primes in their factorization as n, and the units of $\mathbb{Z}_n$ are those numbers which have 1 as their greatest common divisor with n, it follows that the divisors and the units naturally fall apart in a representation based on prime powers. It also follows that multiplication of a unit $u \in U_n$ with any element $v \in \mathbb{Z}_n$ has $\gcd(u \cdot v, n) = \gcd(v, n)$.

A second useful way of representing $v \in \mathbb{Z}_n$ is given in a "residue number system" (e.g. [8, Section 4.7]), where we use the remainders of v with respect to moduli that are prime to each other and $n = n_1 n_2 \cdots n_r$. The most natural way is to use the prime factorization of n, with the $n_i = p_i^{k_i}$, and so

$$\begin{aligned} v &\simeq (v \bmod n_1, v \bmod n_2, \ldots, v \bmod n_r) \\ &\simeq (v_1, v_2, \ldots, v_r)\,. \end{aligned}$$

The Chinese remainder theorem tells us that this representation is unique (whenever the n_i are prime to each other) and we can thus map from v to $(v_1, v_2, \ldots, v_r)$ and back (an algorithm can be found in [1]). Since $\gcd(v, n) \simeq (\gcd(v_1, n_1), \gcd(v_2, n_2), \ldots, \gcd(v_r, n_r))$ in this representation also the units take a special form:

$$u \simeq (u_1, u_2, \ldots, u_r) \in U_n \Leftrightarrow u_i \in U_{n_i}\,,$$

from which the well known $\phi(n) = \prod_i \phi(n_i)$, when the n_i are prime to each other, follows since $|U_n| = \prod_i |U_{n_i}|$.

First we partition U_n using the Chinese remainder theorem and a result from algebra about the structure of U_n in function of smaller cyclic groups.

Theorem 2 (Structure of U_n). *The multiplicative group U_n is isomorphic to the external direct product of groups U_{m_i} where $n = m_1 m_2 \cdots m_r$ is a factorization of n and the m_i are prime to each other*

$$U_n \simeq U_{m_1} \oplus U_{m_2} \oplus \cdots \oplus U_{m_r} \,.$$

Corollary 3. *Every group U_n can be written as an external direct product of r multiplicative cyclic groups M_{n_i} where $n = n_1 n_2 \cdots n_r$ and $r = \kappa(n) + 1$ for $8 \mid n$ or $r = \kappa(n)$ otherwise, and $\kappa(n)$ is the number of unique prime factors of n.*

Proof. A proof can be found in most abstract algebra books (e.g. [7, page 155]) mostly in the proximity of the fundamental theorem of Abelian groups, but since we need some details for prime factors of the form 2^k later on, we sketch the outline.

Consider the group U_n and a prime factorization of $n = p_1^{k_1} p_2^{k_2} \cdots p_{\kappa(n)}^{k_{\kappa(n)}}$. We already know that U_{n_i} with $n_i \nmid 8$ are cyclic (see Corollary 1), so we only need to consider the case where one of the prime factors is 2^k with $k \geq 3$. For such group we can always generate half of the elements by the cyclic subgroup of powers of 5, the other half of the elements can be reconstructed from this subgroup by a simple multiplication

$$U_{2^k} = \{1 \cdot \langle 5 \rangle, (2^{k-1} - 1) \cdot \langle 5 \rangle\} \,.$$

This comes from the isomorphism $U_{2^k} \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_{2^{k-2}}$. We thus split a prime factor $2^k \mid 8$ into two artificial n_i factors, namely 2 and 2^{k-1}, giving cyclic groups of order 2 and 2^{k-2} so that $\phi(2^k) = 2^{k-1} = 2 \cdot 2^{k-2}$. We note that such a power of 2 thus makes an isomorphic copy of half of U_n. This proves the corollary. □

So, given a group U_n we can find r smaller cyclic groups U_{n_i} of order $\phi(n_i)$ and generators g_i (we make abstraction of the special case for $n_i \mid 8$ since it would clutter the explanations following, however the reasoning still holds and an example follows in Section 6.2). We can then order the elements of $\oplus_i U_{n_i}$

in lexicographical order of the powers of these generators

$$\oplus_i U_{n_i} = \left\{ \begin{array}{c} (g_1^0, \ldots, g_{r-1}^0, g_r^0), \ldots, (g_1^0, \ldots, g_{r-1}^0, g_r^{\phi(n_r)-1}), \\ (g_1^0, \ldots, g_{r-1}^1, g_r^0), \ldots, (g_1^0, \ldots, g_{r-1}^1, g_r^{\phi(n_r)-1}), \\ \vdots \\ (g_1^{\phi(n_1)-1}, \ldots, g_{r-1}^{\phi(n_{r-1})-1}, g_r^0), \ldots, (g_1^{\phi(n_1)-1}, \ldots, g_{r-1}^{\phi(n_{r-1})-1}, g_r^{\phi(n_r)-1}) \end{array} \right\},$$

as well as in order of the negative powers of these generators to build a Cayley table. Then it is not so hard to see that the Cayley table of such a group will contain nested circulant matrices.

As an example consider U_n, where $n = p_1 p_2$, a generator g_1 for U_{p_1} and g_2 for U_{p_2}, then the Cayley table can be made to look like:

$$\begin{array}{c|cccc} \cdot & (g_1^0, G_2^{-1}) & (g_1^{-1}, G_2^{-1}) & \cdots & (g_1^1, G_2^{-1}) \\ \hline (g_1^0, G_2) & (g_1^0, C_2) & (g_1^{-1}, C_2) & \cdots & (g_1^1, C_2) \\ (g_1^1, G_2) & (g_1^1, C_2) & (g_1^0, C_2) & \cdots & (g_1^2, C_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (g_1^{-1}, G_2) & (g_1^{-1}, C_2) & (g_1^{-2}, C_2) & \cdots & (g_1^0, C_2) \end{array}, \quad C_2 = \begin{bmatrix} g_2^0 & g_2^{-1} & \cdots & g_2^1 \\ g_2^1 & g_2^0 & \cdots & g_2^2 \\ \vdots & \vdots & \ddots & \vdots \\ g_2^{-1} & g_2^{-2} & \cdots & g_2^0 \end{bmatrix}.$$

Here C_2 is the circulant form of the Cayley table for U_{p_2} and the notation (g_1^k, C_2) means that we have to combine g_1^k with every entry from C_2, likewise the notation (g_1^k, G_2) means to combine every element from $G_2 = \langle g_2 \rangle$ with g_1^k, the same goes for (g_1^{-k}, G_2^{-1}) where $G_2^{-1} = \langle g_2^{-1} \rangle$.

The complete Cayley table for U_n can now be seen as a block circulant matrix with circulant blocks. This can obviously be extended to more than 2 factors and we would get that the Cayley table for r factors could be seen as r nested circulants. A matrix-vector product with such a (block) circulant with r levels can be done in time $O(rn\log(n))$, see Lemma 2 in Section 5 later on in this paper.

4.2 Partitioning of $\mathbb{Z}_n$

We will now split $\mathbb{Z}_n$ by considering a group action φ by the transformation group U_n on $\mathbb{Z}_n$. For completeness we will introduce the necessary definitions.

Definition 4 (Group action, orbit and stabilizer). *A group G acts on a set X by a group action $\varphi : G \times X \to X$ such that for every $x \in X$:*

(1) $\varphi(e, x) = x$, where e is the identity element from G,
(2) $\varphi(g, \varphi(h, x)) = \varphi(gh, x)$ for all $g, h \in G$.

The orbit of $x \in X$ is defined as $\operatorname{orb}(x) = \{\varphi(g, x) : g \in G\}$ and is the subset of X which can be reached by x under the transformations of G. The stabilizer of $x \in X$ is defined as $\operatorname{stab}(x) = \{g \in G : \varphi(g, x) = x\}$ and is the set of transformations which leave x invariant.

For our purpose the group $G = U_n$ acts on the set $X = \mathbb{Z}_n$, and the group action φ is multiplication modulo n. This is in fact a very natural view on the generation of the point set P_n as given in (2), where the components of the generating vector are selected from the units of $\mathbb{Z}_n$, i.e. U_n, to assure that $(k \cdot z_j)/n$ is a permutation of the equispaced distribution k/n in each dimension j, where $k = 0, \ldots, n-1$.

We can now see every possible choice of z_s in the optimization process for dimension s as a possible permutation of the n 1-dimensional points. A given action G on X defines an equivalence relation where the different orbits are the partitions. We will now show that the divisors of n are valid representers for these orbits on $\mathbb{Z}_n$.

Theorem 3 (Partitioning of $\mathbb{Z}_n$). *The union of all orbits generated by the divisors of n under U_n form a partition of $\mathbb{Z}_n$*

$$\bigcup_{d \mid n} \operatorname{orb}(d) = \mathbb{Z}_n \,,$$

where the partition represented by d, and $d \mid n$, has size $|\operatorname{orb}(d)| = \phi(n/d)$. Furthermore

$$n = \sum_{d \mid n} |\operatorname{orb}(d)| \,.$$

Proof. We first note that the orbit of d can be specified in different ways

$$\begin{aligned} \operatorname{orb}(d) &= \{d \cdot u : u \in U_n\}\,, && \text{the definition,} \\ &= d\,U_n\,, && \text{as a coset of } U_n, \\ &= \{v \in \mathbb{Z}_n : \gcd(v, n) = d\}\,, && \text{having a common gcd.} \end{aligned} \tag{6}$$

The last form is the most interesting for us (and follows directly from observing the prime powers). Since for all $v \in \mathbb{Z}_n$

$$\gcd(v, n) = d \in \operatorname{divisors}(n)\,,$$

this shows that the divisors are representers for the partitions.

If we consider $v \in \mathbb{Z}_n$ in the residue number system where the moduli are given by the prime factorization of n, then we can count the number of elements in the orbit of $d \bmod p_i^{k_i}$. The total number of elements in the orbit of d under U_n in $\mathbb{Z}_n$ is then given by their product. Thus for all $u \in U_n$ and a divisor d we consider

$$d \cdot u = (d_1 \cdot u_1, d_2 \cdot u_2, \ldots, d_r \cdot u_r)\,,$$

where the number of elements in the smaller orbit of d_i (under U_{n_i} in $\mathbb{Z}_{n_i}$, i.e. taken modulo n_i where $n_i = p_i^{k_i}$) is given as

$$|\mathrm{orb}_{U_{n_i}}(d_i)| = |d_i\, U_{n_i}| = \begin{cases} \phi(p_i^{k_i}) = |U_{n_i}|\,, & \text{if } \gcd(d_i, p_i^{k_i}) = 1 \text{ (i.e. } d_i \in U_{p_i^{k_i}})\,, \\ \phi(p_i^{k_i-\ell})\,, & \text{if } \gcd(d_i, p_i^{k_i}) = p_i^{\ell}\,. \end{cases}$$

The second case follows since the moduli are of the form p^k, with p prime.

If we now write $p^k \parallel n$ to denote that p^k divides n exactly, i.e. k is the highest power of p that divides n, then it follows that

$$|\mathrm{orb}(d)| = |d\,U_n| = \prod_{\substack{p^k \,\|\, n \\ p^\ell \,\|\, d}} \phi(p^k/p^\ell) \prod_{\substack{p^k \,\|\, n \\ p \nmid d}} \phi(p^k)\,, \qquad \text{for all } d \mid n\,,$$

which simplifies to $|\mathrm{orb}(d)| = \phi(n/d)$. □

The result of the previous theorem contains an instantiation of the very well known identity

$$n = \sum_{d \mid n} \phi(n/d) = \sum_{d \mid n} \phi(d)\,.$$

4.3 Block-partitioning of Ξ_n

If we combine Theorem 2 and Theorem 3 then we can partition Ξ_n in blocks which are isomorphic to the Cayley table of groups $U_{n/d}$. Let us first introduce the following lemma.

Lemma 1. *The following are equivalent for $n = \prod_i n_i$ and all n_i prime to each other:*

$$d\,U_n = d\,U_{n/d} = \oplus_i\, d_i\, U_{n_i/g_i} = d\, \oplus_i\, U_{n_i/g_i}\,, \quad \textit{with } g_i = \gcd(d_i, n_i) \textit{ and } d \mid n\,.$$

Moreover

$$v\,U_n = v\,U_{n/g}\,, \qquad\qquad \textit{with } g = \gcd(v,n) \textit{ and } v \in \mathbb{Z}_n\,.$$

Proof. If $d \mid n$ it follows from the last two equations in (6) that

$$\begin{aligned} d\,U_n &= \{v \in \mathbb{Z}_n : \gcd(v,n) = d\} \\ &= \{v \in d\,\mathbb{Z}_n : \gcd(v,n) = d\} \\ &= \{d \cdot v : v \in \mathbb{Z}_{n/d} \text{ and } \gcd(v,n) = 1\} \\ &= d\,U_{n/d}\,. \end{aligned}$$

To prove that $d\,U_{n/d} = \oplus_i\, d_i\, U_{n_i/g_i}$ we observe that $d_i = d \bmod n_i$ splits in a part that is a unit and a part that is a divisor of n_i

$$d_i = u_i \cdot \tilde{d}_i\,, \qquad\qquad \text{where } \tilde{d}_i \mid n_i \text{ and } u_i \in U_{n_i}\,.$$

We do not care about the unit part because this just gives a permutation of the elements. The effect of multiplying the set U_{n_i} with d_i is thus the same as multiplying with $\tilde{d}_i$. Since $\gcd(d_i, n_i) = \gcd(\tilde{d}_i, n_i) = \tilde{d}_i \bmod n_i$ the result follows.

The same observations can be made for the general case $v\,U_n = v\,U_{n/g}$ with v any number and $g = \gcd(v,n)$ using the Chinese remainder theorem to reconstruct everything back together. □

We should note that we could as well have written $v\,U_n = g\,U_{n/g}$ instead of $v\,U_n = v\,U_{n/g}$, and similar $d\,U_n = \oplus_i\, g_i\, U_{n_i/g_i}$.

In the next theorem we use the symbol $\mathbf{1}_{t\times 1}$ to denote a vector with all components equal to one of length t and we use the symbol $\otimes$ to denote the Kronecker tensor product. So $\mathbf{1}_{t\times 1} \otimes B$ can be read as t replications of the matrix B on top of each other.

Theorem 4 (Block-partitioning of Ξ_n). *We can block-partition the matrix*

$$\Xi_n = \Big[k \cdot z\Big]_{\substack{z \in U_n \\ k \in \mathbb{Z}_n}}$$

by considering the divisors of n (denoted as $d^{(1)}$, $d^{(2)}$, …, $d^{(d(n))}$, with $d(n)$ the number of divisors of n), into vertical partitions A_d per divisor d,

$$\Xi_n = [A_{d^{(1)}} | A_{d^{(2)}} | \cdots | A_{d^{(d(n))}}]\,, \qquad\qquad A_d = \Big[k \cdot z\Big]_{\substack{z \in U_n \\ k \in d\,U_{n/d}}}\,,$$

of sizes $\phi(n) \times \phi(n/d)$ which (each separately) can be horizontally partitioned into t_d identical square blocks B_d for which

$$A_d = \begin{bmatrix} B_d \\ \hline \vdots \\ \hline B_d \end{bmatrix} = \mathbf{1}_{t_d \times 1} \otimes B_d \,, \qquad B_d = \Big[k \cdot z \Big]_{\substack{z \in U_{n/d} \\ k \in d\,U_{n/d}}} = \Big[d \cdot k \cdot z \Big]_{\substack{z \in U_{n/d} \\ k \in U_{n/d}}} ,$$

of size $\phi(n/d) \times \phi(n/d)$, and so $t_d = \phi(n)/\phi(n/d)$, which are isomorphic with the Cayley table of $U_{n/d}$.

Proof. The vertical partitioning follows directly from Theorem 3 and the equality $d\,U_n = d\,U_{n/d}$ from Lemma 1.

What's left to prove is that the vertical partitions can be partitioned again, horizontally, in $t_d = \phi(n)/\phi(n/d)$ identical square blocks which are isomorphic to the Cayley table of $U_{n/d}$. We will again shift our problem to the residue number system, with moduli given by the prime factorization of $n = n_1 n_2 \cdots n_r$. From Lemma 1 we have that

$$d\,U_{n/d} = \oplus_i \, d_i\,U_{n_i/g_i} = d \, \oplus_i \, U_{n_i/g_i} \,, \qquad \text{with } g_i = \gcd(d_i, n_i) \,,$$

and it follows that

$$|d_i\,U_{n_i/g_i}| = |U_{n_i/g_i}| = \phi(n_i/g_i) \,.$$

Now consider the matrix B_{d_i} as a set (i.e. the codomain of B_{d_i})

$$\begin{aligned} B_{d_i} &= \{ d_i \cdot v_i \cdot w_i : v_i \in U_{n_i}, w_i \in U_{n_i/g_i} \} \\ &= \{ d_i \cdot \underbrace{(v_i \cdot w_i)}_{\text{permutation}} : v_i \in U_{n_i}, w_i \in U_{n_i} \} \\ &= \{ d_i \cdot w_i : w_i \in U_{n_i} \} \\ &= \{ (d_i \cdot w_i) : (d_i \cdot w_i) \in d_i\,U_{n_i/g_i} \} \\ &= d_i\,U_{n_i/g_i} \,, \end{aligned}$$

which has size $|B_{d_i}| = \phi(n_i/g_i)$. We observe that the $\phi(n_i)$ elements of U_{n_i} can be partitioned in $t_{d,i} = \phi(n_i)/\phi(n_i/g_i)$ equivalence classes which have the elements of U_{n_i/g_i} as representers, i.e. B_{d_i} is isomorphic to U_{n_i/g_i}.

Using the Chinese remainder theorem it follows that we have $t_d = \phi(n)/\phi(n/d)$ and that B_d is isomorphic to the Cayley table of $U_{n/d}$. □

The previous theorem has not filled in the details of how exactly the rows and the columns of the matrix Ξ_n should be permuted. It only states that this is possible for each vertical partition A_d (cf. the wording *each separately*).

The following corollary states that we can fix these two permutations for the complete matrix at once.

Corollary 4. *If we fix the ordering of the rows for all partitions A_d and arrange the ordering of the columns so that $B_1 \simeq C_n$, where C_n is the (block) circulant form of the Cayley table of the group U_n, then the interleaving of the (block) circulant matrices $B_d \simeq d\,C_{n/d}$ for a certain divisor d of n in A_d is given by*

$$A_d = (\otimes_i R_{d,i})\, d\, C_{n/d}\,,$$

where the i indices go over the prime factors of n (with the exceptional case for powers of 2 as given in Corollary 3) and the R_i matrices are defined as

$$R_{d,i} = \mathbf{1}_{t_{d,i} \times 1} \otimes I_{\phi(n_i/g_i)}\,,$$

where $g_i = \gcd(d_i, n_i)$, $t_{d,i} = \phi(n_i/g_i)/\phi(g_i)$ and $I_{\phi(n_i/g_i)}$ is the identity matrix of order $\phi(n_i/g_i)$.

Proof. By simple calculation. □

This corollary gives a straight forward method to know where every element of a matrix B_d arrives in its corresponding matrix A_d. We just have to interpret what multiplication with a matrix $R_d = \otimes_i R_{d,i}$ means, this interpretation is quite natural since $d\,C_{n/d} = \oplus_i\, d_i\, C_{n_i/g_i}$. If we look at the basic case of multiplication with one matrix $R_{d,i}$ we observe that the identity matrix has the effect of distributing every element of the group $d_i\, C_{n_i/g_i}$, and the replicating by $\mathbf{1}_{t_{d,i}\times 1}$ fills up the empty space left in C_{n_i} when $g_i \neq 1$.

By using the formulation of Corollary 4 we will be able to distribute the results of matrix-vector multiplications with the smaller (block) circulant matrices B_d to the final result vector of multiplication with the complete matrix.

5 Fast matrix-vector for general n

With the result from Theorem 4 it becomes trivial to show that a fast matrix-vector product with matrices homomorph to Ξ_n is possible in time $O(n\log(n))$.

Theorem 5 (Fast matrix-vector for Ξ_n structures). *A matrix-vector product with a matrix Ω_n which is homomorph to Ξ_n can be done in time $O(n\log(n))$ and requires memory of order $O(n)$.*

Proof. We block-partition the matrix Ω_n according to the block-partitioning of Ξ_n given in Theorem 4 by the divisors of n and order the elements appropriately by the powers of the generators of the components of n. This creates vertical partitions A_d which have (interleaved) copies of (block) circulant matrices B_d.

Since the matrix Ω_n is homomorph to Ξ_n the same permutations can be done and we can then assume an arbitrary elementwise mapping function φ on Ξ_n to obtain Ω_n. For our specific purpose this mapping function is $\varphi(t) = \omega(t/n)$. We thus consider the matrix-vector product

$$\begin{aligned}\boldsymbol{E} &= \varphi\left([A_{d^{(1)}}|A_{d^{(2)}}|\cdots|A_{d^{(d(n))}}]\right) \cdot \left[\begin{array}{c} \boldsymbol{p}_{d^{(1)}} \\ \hline \boldsymbol{p}_{d^{(2)}} \\ \hline \vdots \\ \hline \boldsymbol{p}_{d^{(d(n))}} \end{array}\right] \\ &= \varphi(A_{d^{(1)}}) \cdot \boldsymbol{p}_{d^{(1)}} + \varphi(A_{d^{(2)}}) \cdot \boldsymbol{p}_{d^{(2)}} + \cdots + \varphi(A_{p^{(d(n))}}) \cdot \boldsymbol{p}_{d^{(d(n))}} \,,\end{aligned}$$

which can be considered as the sum of $d(n)$ smaller matrix-vector products, where $d(n)$ is the number of divisors of n.

According to Theorem 4 we then have, in each of these A_d matrices, copies of a (block) circulant matrix B_d. As such it suffices to calculate $B_d \cdot \boldsymbol{p}_d$ instead of $A_d \cdot \boldsymbol{p}_d$ for each $d \mid n$ and then distribute and sum the results of these smaller calculations by usage of Corollary 4 and the forthcoming Lemma 3.

The size of B_d is $\phi(n/d) \times \phi(n/d)$ and it follows that a matrix-vector product with B_d can thus be done in $O(\kappa(n/d)\,\phi(n/d)\log(\phi(n/d)))$ (see Lemma 2 following this theorem). The summation of the $d(n)$ smaller result vectors is $O(\kappa(n)\,n)$ (see Lemma 3 following) and this brings the total cost to

$$O\left(\sum_{d\,|\,n} \kappa(n/d)\,\phi(n/d)\log(\phi(n/d)) + \kappa(n)\,n\right) = O(n\log(n))\,, \tag{7}$$

where we used $\sum_{d|n}\phi(d) = n$.

The memory needed for this operation is $O(n)$, since we only need the fast Fourier transform of the first column of the (block) circulant matrices (see Lemma 2). $\square$

In (7) we actually assume $\kappa(n)$ to be bounded by a constant. For practical implementations this will always be the case and will be reasonable, e.g. $\kappa(n) \leq 9$ for all $n \leq 2^{32}$. However, we could as well use a crude bound like $\kappa(n) < \log_2(n)$ (the logarithm in base 2) from which it follows that the total complexity is always less than $O(n\log^2(n))$ (the logarithm to the power 2), this is a serious overestimate however. Also it is known that on the average $\kappa(n) \sim \log(\log(n))$, and for the worst choice, i.e. n a product of distinct primes, it is known that on the average $\kappa(n) \sim \frac{\log(n)}{\log(\log(n))}$.

We have already claimed (and used) that all these block circulant matrices allow for a fast matrix-vector multiplication, no matter how many blocks are embedded. We provide a method for such a fast matrix-vector multiplication in the following lemma.

Lemma 2. *A matrix-vector product with a block circulant matrix C of order n where the blocks can again be block circulant or circulant (at the lowest level) can be done in time $O(kn\log(n))$ requiring memory $O(n)$, where k is the number of circulant levels.*

Proof. In Corollary 2 it was already shown that a matrix-vector product with a circulant matrix can be done in $O(n\log(n))$ using its eigenvalue decomposition which was given in Theorem 1.

Here we will show that any additional block circulant level can be made block diagonal, and furthermore completely diagonal by the use of a permutation and an additional sequence of FFT's (one per diagonal block).

Assume $C = \mathrm{circ}(C^{(0:m-1)})$ is a block circulant matrix with m circulant blocks of order n (i.e. there are 2 levels):

$$C = \mathrm{circ}(C^{(0:m-1)}) = \left[\begin{array}{c|c|c|c|c} C^{(0)} & C^{(m-1)} & C^{(m-2)} & \dots & C^{(1)} \\ \hline C^{(1)} & C^{(0)} & C^{(m-1)} & \ddots & \vdots \\ \hline C^{(2)} & C^{(1)} & \ddots & \ddots & C^{(m-2)} \\ \hline \vdots & \ddots & \ddots & \ddots & C^{(m-1)} \\ \hline C^{(m-1)} & \dots & C^{(2)} & C^{(1)} & C^{(0)} \end{array}\right] \in \mathbb{R}^{nm\times nm},$$

and $C^{(j)} \in \mathbb{R}^{n\times n}$. Then we can diagonalize the smaller circulant matrices by m n-point Fourier transforms on the first column of C (i.e. on the first columns of the blocks $C^{(j)} = \mathrm{circ}(\boldsymbol{c}^{(j)})$). This gives the start of an analog to the diagonalization in Theorem 1 (by applying this same theorem m times):

$$C = (I_m \otimes F_n^{-1}) \cdot \mathrm{circ}(\tilde{C}^{(0:m-1)}) \cdot (I_m \otimes F_n)\,,$$

where $\tilde{C}^{(j)} = \mathrm{diag}(F_n \cdot \boldsymbol{c}^{(j)}) = \mathrm{diag}(\tilde{\boldsymbol{c}}^{(j)})$.

We observe that the elements in $\mathrm{circ}(\tilde{C}^{(0:m-1)})$ are laid out in such a way that we actually have n interleaved circulant matrices of order m, e.g. the first circulant matrix is defined by the first elements of the diagonals $\tilde{\boldsymbol{c}}^{(j)}$, $j = 0, \ldots, m-1$, the second circulant matrix is defined by the second elements, etc... The next step is to exchange the single circ-operation and the m diag-operations with one diag-operation and n circ-operations. This can be done by the following permutation (as can easily be verified by a small example)

$$\mathrm{diag}(\tilde{C}'^{(0:n-1)}) = P_\sigma^T \cdot \mathrm{circ}(\tilde{C}^{(0:m-1)}) \cdot P_\sigma \,,$$

where $\sigma(i) = (i \bmod m)\, n + \lfloor i/m \rfloor$ and $\tilde{C}'^{(k)} = \mathrm{circ}(\tilde{\boldsymbol{c}}_k^{(0:m-1)}) = \mathrm{circ}(\tilde{\boldsymbol{c}}'^{(k)})$.

The diagonalization of the complete matrix C can now be completed by applying n m-point Fourier transforms to these n circulant blocks. As such we find that

$$C = (I_m \otimes F_n^{-1}) \cdot P_\sigma \cdot (I_n \otimes F_m^{-1}) \cdot \tilde{\tilde{C}}' \cdot (I_n \otimes F_m) \cdot P_\sigma^T \cdot (I_m \otimes F_n) \,,$$

with $\tilde{\tilde{C}}' = \mathrm{diag}(F_m \cdot \tilde{\boldsymbol{c}}'^{(0:n-1)})$.

This can be reformulated in a computationally more interesting way when we consider a matrix $\mathcal{C} \in \mathbb{R}^{n \times m}$ defined as

$$\mathcal{C} = \left[\boldsymbol{c}^{(0)} \middle| \boldsymbol{c}^{(1)} \middle| \cdots \middle| \boldsymbol{c}^{(m-1)} \right] \qquad \sim \qquad C = \mathrm{circ}(C^{(0:m-1)}) \,,$$

so that

$$\begin{aligned} \tilde{\mathcal{C}} &= F_n \cdot \mathcal{C} && \sim && \mathrm{circ}(\tilde{C}^{(0:m-1)}) \,, \\ \tilde{\mathcal{C}}' &= \tilde{\mathcal{C}}^T && \sim && \mathrm{diag}(\tilde{C}'^{(0:n-1)}) \,, \end{aligned}$$

and

$$\tilde{\tilde{\mathcal{C}}}' = F_m \cdot \tilde{\mathcal{C}}' \qquad \sim \qquad \tilde{\tilde{C}}' = \mathrm{diag}(F_m \cdot \tilde{\boldsymbol{c}}'^{(0:n-1)}) \,.$$

A matrix-vector product with such a 2-level block circulant matrix has an analog to Corollary 2 as

$$C \cdot \boldsymbol{x} = (I_m \otimes F_n^{-1}) \cdot P_\sigma \cdot (I_n \otimes F_m^{-1}) \cdot \tilde{\tilde{C}}' \cdot (I_n \otimes F_m) \cdot P_\sigma^T \cdot (I_m \otimes F_n) \cdot \boldsymbol{x} \,.$$

From this follows that a matrix-vector product can be done in a fast way by considering $\boldsymbol{x}$ as an $n \times m$ matrix, denoted as $\boldsymbol{x}_{n\times m}$, in column order, so that the product can be calculated efficiently as

$$(C \cdot \boldsymbol{x})_{n\times m} = F_n^{-1} \cdot (F_m^{-1} \cdot (\tilde{\tilde{\mathcal{C}}}' \odot (F_m \cdot (F_n \cdot \boldsymbol{x}_{n\times m})^T)))^T \,,$$

where $\odot$ means elementwise multiplication, e.g. $\text{diag}(\boldsymbol{d})\cdot\boldsymbol{x} = \boldsymbol{d}\odot\boldsymbol{x}$. This calculation has a preprocessing cost of $O(nm\log(nm))$ and a cost of $O(2nm\log(nm))$ per matrix-vector product (ignoring the constants of the FFT's).

If we now consider such matrices C to be embedded in another block circulant with ℓ blocks, and perform all the previous steps for each such matrix C then we arrive at a block circulant with diagonal blocks (at cost $O(\ell nm\log(nm))$). Again we can permute this form to a block diagonal matrix with circulant blocks, perform nm Fourier transforms of size ℓ (at cost $O(nm\ell\log(\ell))$) and we again arrive at a diagonal matrix (at cost $O(nm\ell\log(nm\ell))$).

The matrix-vector product can be handled in exactly the same way as for the 2-level case, with one extra level of FFT's and permutations. The diagonal multiplication is $O(nm\ell)$ and thus the total cost here is as expected $O(nm\ell\log(nm\ell))$.

The total cost for a matrix of size n which has k levels of circulant embeddings thus is $O(kn\log(n))$. The memory needed is the memory needed to store $\mathcal{C}$ and $\boldsymbol{x}$ as well as their k-fold Fourier transforms, this is $O(n)$. $\square$

It must be noted that the number of embedded circulant matrices in a B_d block is the number of distinct prime factors in n/d and thus $k = \kappa(n/d)$ and the size of such a B_d block is $\phi(n/d)$ (again when $n \mid 8$ we have an exceptional case where we have one extra level due to the isomorphic copy effect).

The second part in the total complexity for multiplication with Ω_n is the summing of the $d(n)$ result vectors. We could bound this naively by $d(n)\,\phi(n)$ and for prime n we have $d(p) = 2$ and $\phi(p) = p - 1$, and as such this cost is negligible compared with the $O(n\log(n))$ for the matrix-vector products. Also for prime powers this works out fine, since then $d(p^k) = k+1 = O(\log(n))$ and $\phi(p^k) = p^{k-1}(p-1) = O(n)$. However, for general n this looks not so good.

For composite n we could bound $d(n)\,\phi(n)$ very crudely as

$$d(n)\,\phi(n) \leq 2\sqrt{n}\,(n - \sqrt{n}) = 2(n^{3/2} - n)\,,$$

which will then dominate the cost over the $O(n\log(n))$ from the matrix-vector products. Summing like this will give a total complexity of $O(n^{3/2} - n)$ but is still asymptotically better than $O(n^{2-\delta})$ for doing the full matrix-vector product as shown in Section 2. Luckily it is possible to do a better summing job by carefully choosing the order of the divisors.

Lemma 3. *The summing of the $d(n)$ result vectors of the (block) circulant matrices B_d can be done in time $O(\kappa(n)\,n)$.*

Proof. To achieve a good summing order we consider the divisors of n in natural ordering of their prime components (so the divisors itself appear out of order). For $n = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, with $r = \kappa(n)$, we consider divisors

$$d = p_1^{\ell_1} p_2^{\ell_2} \cdots p_r^{\ell_r}\,, \qquad\qquad \text{where } 0 \le \ell_i \le k_i\,,$$

where we first vary through all the powers of p_1, then increase the power of p_2 by one and vary again through all the powers of p_2 and so on, i.e. we consider the divisors in a "prime power number system" where they can be represented as $d = \langle \ell_r, \ldots, \ell_2, \ell_1 \rangle$ where ℓ_1 is the least significant digit, etc... and our ordering is given by the natural ordering in this number system.

First observe that two adjoining vertical partitions, one for a divisors $d = p_r^{\ell_r} \cdots p_2^{\ell_2} p_1^{\ell_1}$ and the other for a divisor $d' = p_r^{\ell_r} \cdots p_2^{\ell_2} p_1^{\ell_1 + 1}$, need

$$\phi\left(\frac{p_r^{k_r} \cdots p_2^{k_2} p_1^{k_1}}{\gcd(p_r^{\ell_r} \cdots p_2^{\ell_2} p_1^{\ell_1},\, p_r^{\ell_r} \cdots p_2^{\ell_2} p_1^{\ell_1+1})}\right) = \phi(p_r^{k_r-\ell_r} \cdots p_2^{k_2-\ell_2} p_1^{k_1-\ell_1})$$

operations to sum their results. This summing of the adjoining partitions at the lowest level must be done for $\ell_1 = 0, \ldots, k_1 - 1$ and this for all occurrences (i.e. all other powers). If we do this for decreasing ℓ_1 then the formula above holds for all intermediate results and gives a partial cost for the first level of

$$\begin{aligned}
\sum_{\ell_r=0}^{k_r} \cdots \sum_{\ell_2=0}^{k_2} \sum_{\ell_1=0}^{k_1-1} \phi(p_r^{k_r-\ell_r} \cdots p_2^{k_2-\ell_2} p_1^{k_1-\ell_1}) \\
&= \sum_{\ell_r=0}^{k_r} \phi(p_r^{k_r-\ell_r}) \cdots \sum_{\ell_2=0}^{k_2} \phi(p_2^{k_2-\ell_2}) \sum_{\ell_1=0}^{k_1-1} \phi(p_1^{k_1-\ell_1}) \\
&= p_r^{k_r} \cdots p_2^{k_2} (p_1^{k_1} - 1) = O(n)\,.
\end{aligned}$$

We now have result vectors of sizes $\phi(p_r^{k_r-\ell_r} \cdots p_2^{k_2-\ell_2} p_1^{k_1})$, and thus, from now on, the lowest level will always count for its full size $\phi(p_1^{k_1})$. Similarly, to sum up the next level we get a cost of

$$\begin{aligned}
\sum_{\ell_r=0}^{k_r} \cdots \sum_{\ell_2=0}^{k_2-1} \phi(p_r^{k_r-\ell_r} \cdots p_2^{k_2-\ell_2} p_1^{k_1}) &= \sum_{\ell_r=0}^{k_r} \phi(p_r^{k_r-\ell_r}) \cdots \sum_{\ell_2=0}^{k_2-1} \phi(p_2^{k_2-\ell_2})\, \phi(p_1^{k_1}) \\
&= p_r^{k_r} \cdots (p_2^{k_2} - 1)\, \phi(p_1^{k_1}) = O(n)\,.
\end{aligned}$$

And this continues upto the last level, which has a cost of

$$\begin{aligned}
\sum_{\ell_r=0}^{k_r-1} \phi(p_r^{k_r-\ell_r} \cdots p_2^{k_2} p_1^{k_1}) &= \sum_{\ell_r=0}^{k_r-1} \phi(p_r^{k_r-\ell_r}) \cdots \phi(p_2^{k_2})\, \phi(p_1^{k_1}) \\
&= (p_r^{k_r} - 1) \cdots \phi(p_2^{k_2})\, \phi(p_1^{k_1}) = O(n)\,.
\end{aligned}$$

Since we have $r = \kappa(n)$ levels (the number of unique prime factors) and the cost on each level is $O(n)$, the total complexity is $O(\kappa(n)\, n)$. $\square$

6 Illustrative examples

We now provide some examples which show the complete track of the previous sections in action. We will start with the trivial case for prime n. Since powers of 2 are an exceptional application we present such an example which also illustrates the concepts for other prime powers. Finally we will consider an example of the more general case.

We present images of the matrix Ξ_n in Figures 1–3. Since in such a matrix we have values from 0 upto $n-1$, there are n different colors per figure. On each row each color occurs only once, i.e. each row is a permutation of the first row of n colors. The matrices are organized as in Theorem 4 and so there is a vertical partition A_d for each divisor d of n, grouped as given by Lemma 3. The start of these partitions is marked with an arrow which has a textlabel to denote which d generates this partition. In each vertical partition we have repetitive blocks B_d, the repetitions of this block are drawn with faded colors to make clear how this block is distributed (cf. Corollary 4).

6.1 Prime n

Our previous result for prime n from [10] can now compactly be restated as follows. If n equals a prime p, the divisors are simply

$$\operatorname{divisors}(p) = \{1, p\}\,.$$

We obtain the trivial partition $\mathbb{Z}_p = 1\,U_p \cup p\,U_1 = U_p \cup \{0\}$. The index-matrix has thus two very simple vertical partitions A_1 and A_p:

$$\begin{aligned} A_1 &= \mathbf{1}_{1\times 1} \otimes B_1\,, & B_1 &= [k \cdot z]_{z,k \in U_p}\,, \\ A_p &= \mathbf{1}_{p-1\times 1} \otimes B_p\,, & B_p &= [0]\,, \end{aligned}$$

of which B_1 is isomorphic to a circulant matrix of size $\phi(n)$.

For example consider $p = 5$ and the generator $g = 2$ for U_5. Then we can construct A_1 and A_5 as

$$A_1 = B_1 = [1,2,4,3]^T \cdot [1,3,4,2] = \begin{bmatrix} 1 & 3 & 4 & 2 \\ 2 & 1 & 3 & 4 \\ 4 & 2 & 1 & 3 \\ 3 & 4 & 2 & 1 \end{bmatrix}, \qquad A_5 = \mathbf{1}_{4\times 1} \otimes [0] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The diagonal form of $\omega(A_1/n)$ is simply obtained as $\operatorname{diag}(F_4 \cdot \omega([1,2,4,3]^T/5))$.

An example of the structure for $p = 41$ is given in Figure 1. On the left we have drawn the matrix in its natural ordering, while on the right the matrix is drawn in the ordering which allows a fast matrix-vector product. The last partition is the partition for $d = 41$ were the complete column is constant; only the first element in this column is drawn at full color, the rest of the column is faded to denote its redundancy.

6.2 Powers of 2 *(and other prime powers)*

For powers of a prime the divisors are

$$\operatorname{divisors}(p^k) = \{p^\ell : 0 \le \ell \le k\}\,,$$

resulting in circulant matrices B_d (being block circulant with 2 levels for a power of 2), which have regularly diminishing sizes

$$\begin{aligned} |B_{p^\ell}| &= \phi(p^{k-\ell}) \\ &= \begin{cases} p^{k-\ell-1}(p-1)\,, & \text{if } \ell < k\,, \\ 1\,, & \text{otherwise.} \end{cases} \end{aligned}$$

Figure 2 shows Ξ_n for $n = 2^6 = 64$. In this figure it is clearly visible that increasing powers of a prime have the effect of overlapping in a nice way. It is this effect we are using to sum the result vectors. Assume we have calculated the 7 result vectors $\boldsymbol{E}_d$ for $n = 64$, then the summing order from Lemma 3 gives

$$\boldsymbol{E} = \underbrace{\boldsymbol{E}_1 + \underbrace{(\boldsymbol{E}_2 + \underbrace{(\boldsymbol{E}_4 + \underbrace{(\boldsymbol{E}_8 + \underbrace{(\boldsymbol{E}_{16} + \underbrace{(\boldsymbol{E}_{32} + \boldsymbol{E}_{64})}_{\text{1 addition}})}_{\text{2 addition}})}_{\text{4 additions}})}_{\text{8 additions}})}_{\text{16 additions}}}_{\text{32 additions}}$$

resulting in a total of $1 + 2 + 4 + 8 + 16 + 32 = 63 = 64 - 1$ which can be verified on the figure.

Also visible in the figure is the isomorphic copy effect when a power of 2 is involved (mentioned in Corollary 3). This can be used nicely since the kernel function $\omega(\cdot)$ is in most cases symmetric around $1/2$, i.e. $\omega(x) = \omega(1-x)$. As was shown in [10, Theorem 2], for prime n this symmetry has the effect that only half the space of z-candidates has to be searched and only one quarter of the Ω_n matrix has to be considered. Similar effects occur for n which are not prime, for a power of 2 this means that the isomorphic copy can be left out, in other words, we get circulant matrices instead of block circulant matrices for free.

6.3 For general n

To get a better view of the interleaving of the matrices B_d as given in Corollary 4 we must of course consider more general n. Figure 3 shows Ξ_n for $n = 3 \cdot 5 \cdot 7 = 105$. Clearly visible in the part for $d = 1$ is the block circulant structure with 3 levels. At the highest level we see a circulant structure with $\phi(3) = 2$ blocks of size $\phi(5)\phi(7) = 24$, in each such block we see again a circulant structure with $\phi(5) = 4$ blocks of size $\phi(7) = 6$, and these blocks in their turn are circulant matrices of 6×6.

The summing as given by Lemma 3 here gives:

$$\begin{aligned}
\boldsymbol{E} &= \underbrace{(\boldsymbol{E}_1 + \boldsymbol{E}_3)}_{\text{48 additions}} + \underbrace{(\boldsymbol{E}_5 + \boldsymbol{E}_{15})}_{\text{12 additions}} + \underbrace{(\boldsymbol{E}_7 + \boldsymbol{E}_{21})}_{\text{8 additions}} + \underbrace{(\boldsymbol{E}_{35} + \boldsymbol{E}_{105})}_{\text{2 additions}} \\
&= \underbrace{(\boldsymbol{E}_{1,3} + \boldsymbol{E}_{5,15})}_{\text{48 additions}} + \underbrace{(\boldsymbol{E}_{7,21} + \boldsymbol{E}_{35,105})}_{\text{8 additions}} \\
&= \underbrace{(\boldsymbol{E}_{1,3,5,15} + \boldsymbol{E}_{7,21,35,105})}_{\text{48 additions}}
\end{aligned}$$

which makes a total of $(48 + 12 + 8 + 2) + (48 + 8) + (48) = (7 \cdot 5 \cdot (3 - 1)) + (7(5 - 1)\phi(3)) + ((7 - 1)\phi(5)\phi(3)) = 174$ additions and can again be verified on the figure.

7 Discussion

In [4] and [5], Dick & Kuo presented an adaptation of the component-by-component algorithm, called 'Partial search', by using low composite n having 2, 3, 4 and 5 factors. In their adaptation the worst-case error for each of the factors is calculated in terms (by averaging over the components to be optimized) and then the optimal z_i are combined using the Chinese remainder theorem. From our analysis we expect that their calculated errors differ from the true worst-case error probably only in a small amount, since the divisors they left out are large and thus would only give small blocks B_d.

Also from their papers [4,5], with verification in our previous paper [10], and

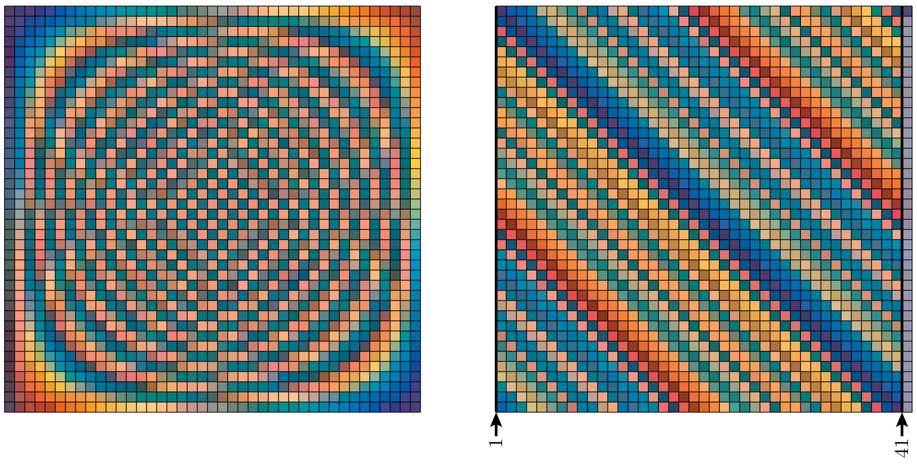

Fig. 1. Example matrix for $n = 41$, left: natural order, right: generator order

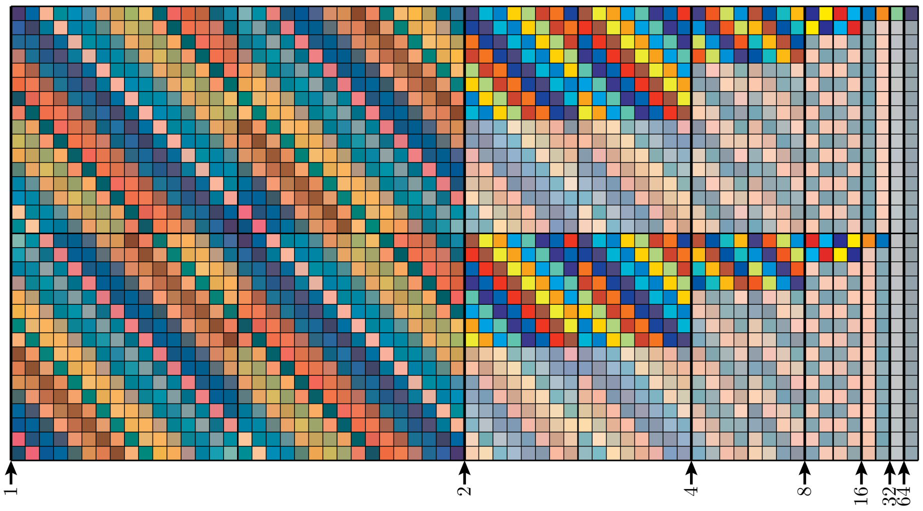

Fig. 2. Example matrix for $n = 2^6 = 64$

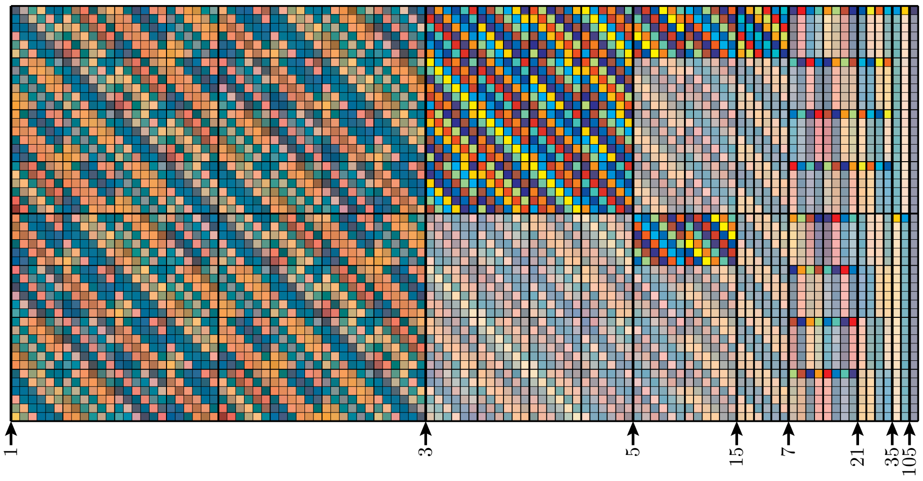

Fig. 3. Example matrix for $n = 3 \cdot 5 \cdot 7 = 105$

also from [3], it is known that prime n have lower worst-case errors and thus are preferable over composite n. This clearly lowers the interest in implementing such a general n routine as presented in this paper. However, the prime power case is certainly interesting as this might give the user the opportunity to apply only part of the pointset while still keeping a good distribution of the used points.

In our opinion, the main contribution of this paper is the revealing of the structure present in rank-1 lattice rules. We expect that it is be possible to get some insights on the effect of combining two existing lattice rules into a new one. Also the 'natural' division of the matrix in blocks associated with the divisors of n could be useful. An important property that will probably be of use here is the possibility of using the Chinese remainder theorem to combine units of different groups whenever their n_i are prime to each other.

We mentioned in the examples section that the kernel function $\omega(\cdot)$ is often symmetric. For computational efficiency an implementation should definitely use this fact because, although the algorithm is called fast, there is a huge difference in waiting 10 minutes for circa 10^8 points (a result from [10]) or waiting more than half an hour...

Only preliminary testing has been done for more general n, but from the results in this paper it should be clear that an implementation for prime n will probably be the most efficient (contradicting the previous results from [3–5] due to the fast construction). Especially n which have a large number of unique prime factors will slow the algorithm down because of the more complicated (block) circulant matrix-vector calculations involved. An example implementation for prime n was given in [9] (as well as a connection with component-by-component construction of polynomial lattice rules).

We conclude that we presented a method to construct rank-1 lattice rules in a weighted, shift-invariant tensor product reproducing kernel Hilbert space by using the component-by-component algorithm and the intrinsic structure present in this setting. The construction has time complexity $O(sn\log(n))$ for a rank-1 lattice rule with n points in s dimensions, for any n, and needs memory $O(n)$. The time complexity increases when n has more unique prime factors and when n has more divisors, but still is $O(sn\log(n))$.

Acknowledgements

The authors would like to thank the organizers of the Dagstuhl Seminar 04401 titled "Algorithms and Complexity for Continuous Problems" for such a stimulating environment to present this work. We would also like to thank Frances

Kuo for renewing our attention to the powers of 2 case. This research is part of a project financially supported by the Onderzoeksfonds K.U.Leuven / Research Fund K.U.Leuven.

References

[1] Henry Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics. Springer-Verlag, 3rd edition, 1996.

[2] Philip J. Davis. *Circulant Matrices*. Wiley, 1979.

[3] Josef Dick. On the convergence rate of the component-by-component construction of good lattice rules. *Journal of Complexity*, 20(4):493–522, August 2004.

[4] Josef Dick and Frances Kuo. Constructing good lattice rules with millions of points. In Harald Niederreiter, editor, *Monte-Carlo and quasi-Monte Carlo Methods - 2002*, pages 181–197. Springer-Verlag, January 2004.

[5] Josef Dick and Frances Kuo. Reducing the construction cost of the component-by-component construction of good lattice rules. *Mathematics of Computation*, 73:1967–1988, 2004.

[6] Matteo Frigo and Steven G. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proc. 1998 IEEE Intl. Conf. Acoustics Speech and Signal Processing, Vol. 3*, pages 1381–1384. IEEE, 1998.

[7] Joseph A. Gallian. *Contemporary Abstract Algebra*. Houghton Mifflin, 4th edition, 1998.

[8] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, 2nd edition, 1994.

[9] Dirk Nuyens and Ronald Cools. Fast component-by-component construction, a reprise for different kernels. In Harald Niederreiter and Denis Talay, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer-Verlag. Accepted.

[10] Dirk Nuyens and Ronald Cools. Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces. *Mathematics of Computation*, 2005. Accepted.

[11] Ian H. Sloan and Andrew V. Reztsov. Component-by-component construction of good lattice rules. *Mathematics of Computation*, 71(237):263–273, 2002.

[12] Ian H. Sloan and Henryk Woźniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals. *Journal of Complexity*, 14(1):1–33, March 1998.