CHALMERS
UNIVERSITY OF TECHNOLOGY

The Voronoi Region of the Barnes-Wall Lattice Λ16

Citation for the original published paper (version of record):
Pook-Kolb, D., Agrell, E., Allen, B. (2023). The Voronoi Region of the Barnes-Wall Lattice Λ16. IEEE Journal on Selected Areas in Information Theory, 4: 16-23.
http://dx.doi.org/10.1109/JSAIT.2023.3276897

(article starts on next page)

The Voronoi Region of the Barnes–Wall Lattice Λ_{16}

Daniel Pook-Kolb, Erik Agrell, *Fellow, IEEE*, and Bruce Allen, *Member, IEEE*

***Abstract*—We give a detailed description of the Voronoi region of the Barnes–Wall lattice Λ_{16}, including its vertices, relevant vectors, and symmetry group. The exact value of its quantizer constant is calculated, which was previously only known approximately. To verify the result, we estimate the same constant numerically and propose a new very simple method to quantify the variance of such estimates, which is far more accurate than the commonly used jackknife estimator.**

***Index Terms*—Barnes–Wall lattice, lattice quantizer, normalized second moment, quantizer constant, Voronoi region**

I. Introduction

In 1959, E. S. Barnes and G. E. Wall introduced a family of lattices in dimensions 2^m, $m \geq 2$, based on Abelian groups [1]. In dimensions 4 and 8, the proposed construction reproduced known lattices, which are nowadays denoted as D_4 and E_8, respectively, whereas previously unknown lattices were obtained in dimensions 16 and up. Alternative constructions and further properties of the *Barnes–Wall (BW) lattices* were investigated in [2]–[4]. A particularly elegant construction is given in [5].

The BW lattices are remarkably good in three of the standard figures of merit for lattices: *packing, kissing,* and *quantization.* In fact, they are known or conjectured to be optimal in all three figures of merit in dimensions $n = 4$, 8, and 16 [6, Ch. 1]. A recent breakthrough proves that the BW lattice E_8 is the optimal 8-dimensional sphere packing [7]. For this reason, they have been applied in a number of applications, including digital communications [2], data compression [8], cryptography [9], quantum error correction [5], [10], and algebraic geometry [11].

The Voronoi regions of D_4 and E_8 have been fully determined. Hence their *packing densities, kissing numbers,* and *quantizer constants* are known exactly [12], [6, Ch. 4], and we will not discuss these lattices further. In this paper, we determine the Voronoi region of the 16-dimensional BW lattice Λ_{16}. Its relevant vectors, vertices and quantizer constant are reported exactly for the first time. We furthermore characterize its full symmetry group, which is known to be of order 89 181 388 800 [6, Section 4.10], using two transformation matrices.

D. Pook-Kolb and B. Allen are with the Max Planck Institute for Gravitational Physics (Albert Einstein Institute), 30167 Hannover, Germany, and Leibniz Universität Hannover (e-mail: daniel.pook.kolb@aei.mpg.de and bruce.allen@aei.mpg.de).

E. Agrell is with the Department of Electrical Engineering, Chalmers University of Technology, 41296 Gothenburg, Sweden (e-mail: agrell@chalmers.se).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material includes a text catalog of the face classes of Λ_{16}. Contact daniel.pook.kolb@aei.mpg.de for further questions about this work.

II. The Face Hierarchy

In this section, we describe the Voronoi region of Λ_{16} in a bottom-up manner, beginning from the 0-faces (vertices) and making our way upwards in the hierarchy of dimensions to the single 16-face, which is the Voronoi region itself. We describe the faces in the coordinate system defined by the lower block triangular generator matrix

$$\left[\begin{array}{cccccccc:cccccccc}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}
\end{array}\right] . \quad (1)$$

This generator matrix is scaled down by a linear factor of $\sqrt{2}$ (or, equivalently, a volume factor of 256) compared with the generator matrix for the same lattice in [6, Fig. 4.10]. Some lattice parameters depend on the scaling of the lattice.

A. 0-faces

The Voronoi region has 201 343 200 vertices, which belong to six equivalence classes listed as $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_6$ in Tab. I. Equivalence is defined by the rotations $\mathrm{Aut}(\Lambda_{16})$ that take Λ_{16} into Λ_{16}. If translation by a lattice vector is considered as another equivalence operation, $\boldsymbol{v}_2$ becomes equivalent to $\boldsymbol{v}_4$ and $\boldsymbol{v}_3$ to $\boldsymbol{v}_5$, reducing the six equivalence classes to only four. The vertices are located at a squared distance from the origin of $3/2$, $10/9$, or 1. Hence, the covering radius is $\sqrt{3/2}$, as already known [3], [6, Section 4.10].

B. 1-faces

The vertices are connected by a total of about $3 \cdot 10^{10}$ edges, which belong to 23 equivalence classes.[1] Their lengths are $\sqrt{3/2}$, 1, $\sqrt{17/18}$, $\sqrt{7/3}$, $\sqrt{11/18}$, $2/3$, $\sqrt{5/18}$, and

[1] We explicitly construct at least one representative per equivalence class. However, we do not create the full orbits of those representatives. The number of faces is instead obtained by estimating the orbit sizes using [13].

TABLE I: Representatives of the relevant vectors $\boldsymbol{n}_i$ (first two rows) and vertices $\boldsymbol{v}_i$ (remaining six rows) of the Voronoi region of Λ_{16} in order of increasing length. Shown are the components, squared lengths, sizes of the orbits under $\mathrm{Aut}(\Lambda_{16})$, and sizes of the respective stabilizer subgroups of $\mathrm{Aut}(\Lambda_{16})$.

vector	components	$\|\cdot\|^2$	orbit size	stabilizer size
$\boldsymbol{n}_1$	$(1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$	2	4 320	20 643 840
$\boldsymbol{n}_2$	$1/2\ (2\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ -1\ 0)$	3	61 440	1 451 520
$\boldsymbol{v}_1$	$1/2\ (1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1)$	1	4 320	20 643 840
$\boldsymbol{v}_2$	$1/12\ (9\ 3\ 3\ 3\ 1\ -1\ 1\ 1\ 1\ -1\ 3\ -1\ -3\ 3\ -3\ -3)$	10/9	66 355 200	1 344
$\boldsymbol{v}_3$	$1/6\ (5\ 1\ -1\ -1\ -1\ 1\ 1\ -1\ 1\ -1\ 1\ -1\ 1\ 1\ 1\ 1)$	10/9	2 211 840	40 320
$\boldsymbol{v}_4$	$1/6\ (5\ 1\ 1\ 1\ -1\ 1\ -1\ -1\ 1\ -1\ -1\ -1\ -1\ -1\ 1\ -1)$	10/9	66 355 200	1 344
$\boldsymbol{v}_5$	$1/6\ (5\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1\ -1\ 1\ -1\ -1\ 1)$	10/9	66 355 200	1 344
$\boldsymbol{v}_6$	$1/4\ (3\ 1\ 1\ 1\ 1\ 1\ 1\ -1\ 1\ 1\ 1\ -1\ 1\ -1\ -1\ -1)$	3/2	61 440	1 451 520

$1/3$. At each vertex equivalent to $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$, $\boldsymbol{v}_4$, $\boldsymbol{v}_5$, or $\boldsymbol{v}_6$, respectively, 32 768, 144, 403, 179, 220, or 398 824 edges meet.

C. 2-faces

There are about $5 \cdot 10^{11}$ 2-faces in 58 equivalence classes. These consist of 3 classes of about $4 \cdot 10^9$ squares with an area of $1/9$ and about $5 \cdot 10^{11}$ triangles in 55 classes, 21 of which are geometrically distinct, with areas between $\sqrt{15}/72$ and $\sqrt{5}/4$. The 22 geometrically distinct 2-faces are shown in Fig. 1.

D. 3- to 14-faces

In dimensions 3 to 14, there are 6 052 classes of faces, which we will not describe in detail here. Some of their properties are summarized in Tab. II, where we show the number of face classes under $\mathrm{Aut}(\Lambda_{16})$, numbers of child faces (i.e., subfaces of dimension $d-1$) and vertices for the faces in all dimensions $d = 0, 1, \ldots, 16$. Further information is available as supplementary material [14].

E. 15-faces

The 15-faces, or *facets,* all lie halfway between the origin and another lattice vector, orthogonal to the line between them. There are in total 65 760 such facet-defining nonzero vectors, or *relevant vectors.* They belong to two equivalence classes at different distances from the origin (see Tab. I). The ones closest to the origin are the *minimal vectors* at a squared distance of 2, which were found already in [1]. The packing radius is half of their length, i.e., $\sqrt{2}/2$. There are 4 320 such vectors, which is the kissing number of the lattice. There are also 61 440 other relevant vectors, which have a squared length of 3.

The facets belonging to the 4 320 minimal vectors each have 7 704 child faces and 1 046 430 vertices of all six classes, while the remaining 61 440 facets have 828 child faces and 26 160 vertices equivalent to either $\boldsymbol{v}_2$, $\boldsymbol{v}_4$, $\boldsymbol{v}_5$, or $\boldsymbol{v}_6$.

F. 16-face

Having enumerated all inequivalent d-faces for $d = 0, 1, \ldots, 15$ and computed their volumes and second moments using the recursion relations in [15, Sec. 3], a complete characterization of the 16-face is obtained. Using [13], we estimate that the Voronoi region has between $1 \cdot 10^{14}$ and $3 \cdot 10^{14}$ faces across all dimensions.

Next, the *covariance matrix* or *second moment tensor* is computed as

$$\boldsymbol{U} = \frac{U}{16} \boldsymbol{I}_{16} \,, \tag{2}$$

where the (unnormalized) *second moment*

$$U = \operatorname{tr} \boldsymbol{U} = \frac{207\,049\,815\,983}{4\,287\,303\,820\,800} \tag{3}$$

and $\boldsymbol{I}_{16}$ is the 16×16 identity matrix. After proper normalization, the quantizer constant is obtained as

$$G = \frac{1}{n} \frac{U}{V^{1+2/n}} \,, \tag{4}$$

where $n = 16$ is the lattice's dimension and $V = 1/16$ is the volume of its Voronoi region, which yields

$$G = U\sqrt{2} \approx 0.068\,297\,622\,489\,318\,7 \,. \tag{5}$$

To verify our enumeration of face classes, we use the recursion relations in [15, Sec. 3] to calculate the volume of the Voronoi region, which agrees with the expected value of $1/16$. We also verify the result (5) numerically in Sec. IV.

III. The symmetry group of Λ_{16}

The symmetries of Λ_{16} are generated by products of sign changes, permutations and the matrix

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_4 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_4 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{H}_4 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{H}_4 \end{bmatrix} \,, \tag{6}$$

where

$$\boldsymbol{H}_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{7}$$

is a Hadamard matrix.

There are 2 048 sign changes, which can be described as a product of three subgroups $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. The first subgroup $\mathcal{S}_1$ contains all even numbers of sign changes of component pairs $(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ for $i = 1, 3, \ldots 15$, and has order 128. $\mathcal{S}_2$ changes the signs of an even number of the first and last 4 odd components $(\boldsymbol{x}_i, \boldsymbol{x}_{16-i})$, $i = 1, 3, 5, 7$. This subgroup has

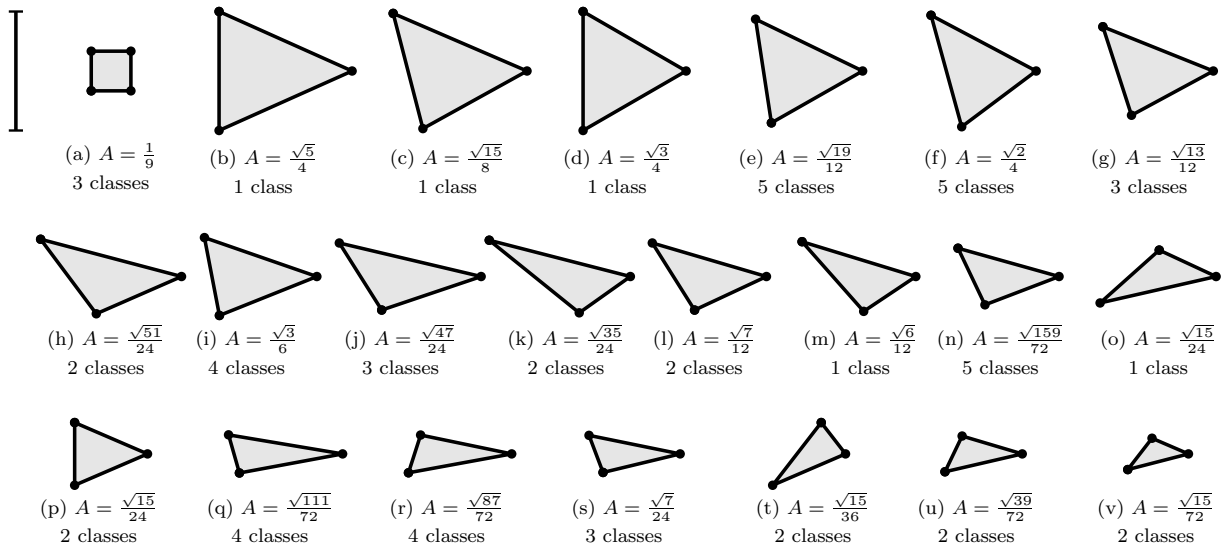


Fig. 1: The 22 geometrically distinct types of 2-faces of the Voronoi region of Λ_{16}. We show the area A as well as the number of classes the face types fall into under $\mathrm{Aut}(\Lambda_{16})$. For reference, the line in the top-left corner has unit length. (a) is a square with edges of length $1/3$. The triangle (d) is equilateral, while (b), (c), (e), (g), (h), (l), (p), (t), and (v) are isosceles. All internal angles of the triangles are strictly less than $90°$. The maximum $\theta_{\max}$ is only found in (n), where $\cos\theta_{\max} = \sqrt{10}/40$ ($\theta_{\max} \approx 85.47°$). The smallest angle satisfies $\cos\theta_{\min} = 29\sqrt{238}/476$ ($\theta_{\min} \approx 19.97°$) and is only found in triangle (q).

order 8. Finally, $\mathcal{S}_3$ is of order 2 and changes the signs of the components $(\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_5, \boldsymbol{x}_7)$.

The permutations $\mathcal{P} \subset \mathrm{Aut}(\Lambda_{16})$ of vector components that keep Λ_{16} invariant are described in [1, Lemma 3.2]. The Lemma makes use of a 4-dimensional vector space over the Galois field $\mathrm{GF}(2)$ to represent indices of components of the lattice vectors. The reader is referred to [1] for a detailed description of this construction. Using [16, Eq. (19) of Ch. 13] the order of $\mathcal{P}$ is

$$|\mathcal{P}| = 16 \prod_{l=0}^{3} (16 - 2^l) = 322\,560 \,, \tag{8}$$

These are precisely the permutations that keep the first-order binary Reed–Muller codes of length 2^4 invariant [16, Theorem 24 of Ch. 13].

Examples of permutations in $\mathcal{P}$ are

$$\begin{aligned} p_1 &= (1\ 2\ 3\ 4)(5\ 6\ 7\ 8)(9\ 10\ 11\ 12)(13\ 14\ 15\ 16) \,, \\ p_2 &= (1\ 2)(5\ 6)(9\ 10)(13\ 14) \,, \\ p_3 &= (1\ 6\ 13)(2\ 8)(3\ 9\ 12\ 5\ 15\ 14)(4\ 11\ 7) \,, \\ p_4 &= (1\ 9\ 16\ 15\ 5\ 7\ 4\ 8\ 10\ 6\ 13\ 2\ 3\ 14\ 12) \,, \end{aligned} \tag{9}$$

here given in cycle notation for compactness. The complete subgroup $\mathcal{P}$ can be generated using various subsets of these permutations, for example $\{p_1, p_2, p_3\}$, $\{p_1, p_4\}$, or $\{p_3, p_4\}$.

The full automorphism group $\mathrm{Aut}(\Lambda_{16})$ can be generated by combining $\boldsymbol{H}$ with the generators of $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ and one of the sets of generators of $\mathcal{P}$. Remarkably, it can also be generated by just two matrices. The first is the 16×16 permutation matrix $\boldsymbol{M}_1$ corresponding to p_3. The second is a matrix

$$\boldsymbol{M}_2 = \begin{bmatrix} \bar{\boldsymbol{H}}_4 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{H}}_4 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \bar{\boldsymbol{H}}_4 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \bar{\boldsymbol{H}}_4 \end{bmatrix} , \tag{10}$$

which is built using (7) with a sign change of the last row, i.e., with the Hadamard matrix

$$\bar{\boldsymbol{H}}_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & 1 & -1 \end{bmatrix} . \tag{11}$$

IV. Numerical Verification and Error Estimates

To validate (3), we estimate U by Monte-Carlo integration over the Voronoi region. We also estimate the variance of the estimate of U, for which we use a different method than the "jackknife estimator" in [17]. In this section, we first describe our estimate of U and the variance thereof, then motivate why we prefer our variance estimator over the jackknife, and finally compare our numerical estimate of G for Λ_{16} with the true value in (5).

The Monte-Carlo estimate of U is

$$\hat{U} = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i\|^2 \,, \tag{12}$$

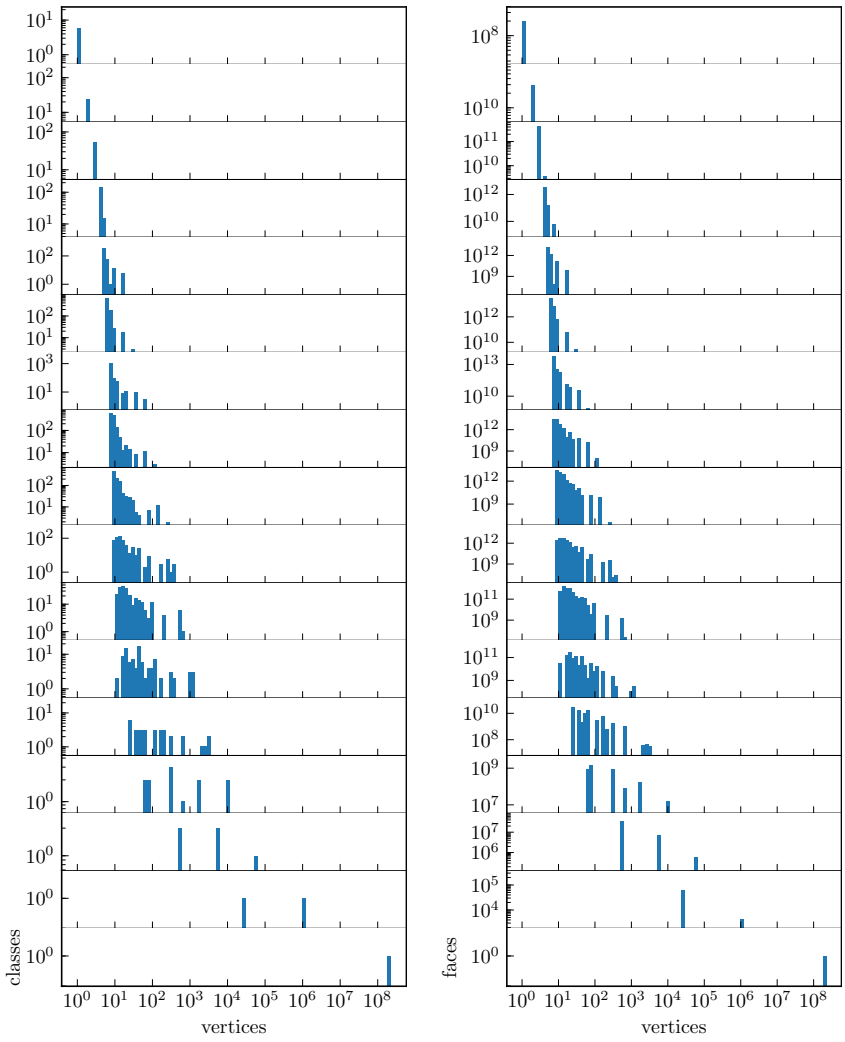
where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ are N independent random vectors uniformly distributed in the Voronoi region of Λ.

To estimate $\mathrm{var}\,\hat{U}$, we first note that since the vectors $\boldsymbol{x}_i$ are independent and identically distributed, $\mathrm{var}\,\hat{U} =$

Equation (12) is incorrect, following a similar error in [17]. Corrections:
* The right-hand side of (12) should be multiplied by V
* $\hat{U}$ in (13)–(15) (but not (16)–(17)) should be replaced by $\hat{U}/V$.

TABLE II: Summary information about the faces of the Voronoi region of Λ_{16}. The first column lists the dimension d of the faces and the second the number of classes of d-faces under $\mathrm{Aut}(\Lambda_{16})$. The third column shows the range of numbers of child faces of each d-face and the fourth column the range of numbers of vertices of each d-face. In the fifth column, we visualize the number of face classes (y-axis) containing a certain number of vertices (x-axis). The last column shows the same information for the numbers of faces instead of face classes, which have been approximated using [13].

dim	classes	child faces	vertices	vertex counts of face classes	vertex counts of faces (approx.)
0	6	0	1		
1	23	2	2		
2	58	3, 4	3, 4		
3	168	4–6	4–8		
4	441	5–16	5–16		
5	867	6–21	6–32		
6	1 257	7–30	7–64		
7	1 329	8–51	8–128		
8	1 023	9–128	9–256		
9	566	10–194	10–400		
10	253	11–258	11–641		
11	96	12–620	12–1 281		
12	35	16–862	24–2 945		
13	12	42–1 312	64–11 138		
14	5	144–2 763	520–59 907		
15	2	828, 7 704	26 160, 1 046 430		
16	1	65 760	201 343 200		



$(1/N)\,\mathrm{var}\|\boldsymbol{x}\|^2$, where $\boldsymbol{x}$ is a single random vector with the same distribution as $\boldsymbol{x}_i$. Therefore, our estimate of $\mathrm{var}\,\hat{U}$, denoted by $\widehat{\mathrm{var}}\,\hat{U}$, is defined by

$$\widehat{\mathrm{var}}\,\hat{U} = (1/N)\,\widehat{\mathrm{var}}\|\boldsymbol{x}\|^2\,. \tag{13}$$

Applying the standard unbiased variance estimator of $\mathrm{var}\|\boldsymbol{x}\|^2$

$$\widehat{\mathrm{var}}\|\boldsymbol{x}\|^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(\|\boldsymbol{x}_i\|^2 - \hat{U}\right)^2 \tag{14}$$

in (13) yields

$$\begin{aligned}\widehat{\mathrm{var}}\,\hat{U} &= \frac{1}{N(N-1)}\sum_{i=1}^{N}\left(\|\boldsymbol{x}_i\|^2 - \hat{U}\right)^2 \\ &= \frac{1}{N-1}\left(\frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i\|^4 - \hat{U}^2\right)\end{aligned} \tag{15}$$

or after normalization as in (4)

$$\hat{G} = \frac{\hat{U}}{nV^{1+2/n}}, \tag{16}$$

$$\widehat{\mathrm{var}}\,\hat{G} = \frac{\widehat{\mathrm{var}}\,\hat{U}}{(nV^{1+2/n})^2}\,. \tag{17}$$

The variance estimator (15) follows directly from fundamental laws of probability. What is surprising is that a different estimator has been used, unchallenged, in most, or perhaps all, previous works involving numerical estimates of lattice second moments [17]–[19]. To rectify this 39-year old misconception, we now elaborate on why (15) is more accurate.

The jackknife works by partitioning the independent randomly selected vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ into g groups, computing the average squared length within each group, and finally computing the sample variance of these g averages [17, Eqs. (3)–

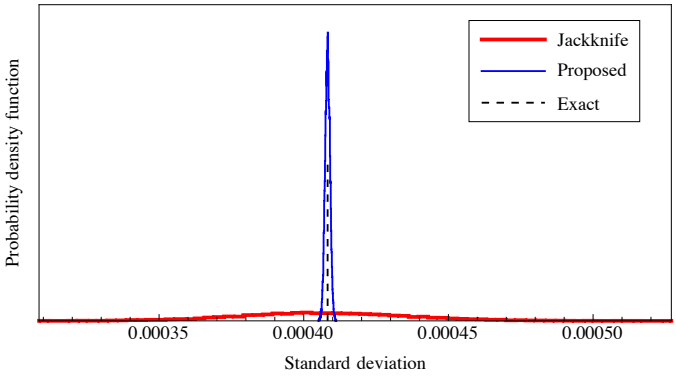


Fig. 2: Histograms of two estimates of the standard deviation of the estimated second moment $\hat{U}$ of the cubic lattice. The exact standard deviation $(\operatorname{var}\hat{U})^{1/2}$, which can be calculated analytically for the cubic lattice, reveals that the proposed estimator (12) is much more accurate than the jackknife with 100 groups.

(4)]. This method brings at least two disadvantages: First, the estimated variance depends on how the list $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is ordered; reordering the list would yield a different variance estimate, although the estimated second moment (12) remains the same. And second, the variance of vectors within a group is ignored. The proposed estimator (15) suffers from neither of these disadvantages.

To quantify the accuracy of both variance estimators, we numerically estimate the second moment of the cubic lattice $\mathbb{Z}^n$ for $n = 3$. The second moment of $\mathbb{Z}^n$ is $U = \mathbb{E}[\|\boldsymbol{x}\|^2] = n/12$, and the variance of $\hat{U}$ can be calculated exactly as $\operatorname{var}\hat{U} = (1/N)\operatorname{var}\|\boldsymbol{x}\|^2 = (1/N)(\mathbb{E}[\|\boldsymbol{x}\|^4] - \mathbb{E}[\|\boldsymbol{x}\|^2]^2) = n/(180N)$. We generated $N = 100\,000$ vectors uniformly in the Voronoi region of $\mathbb{Z}^3$, which is the unit cube, computed $\hat{U}$ using (12), and estimated the variance of $\hat{U}$ using the two methods. For the jackknife, we used a group size of $g = 100$ as in [17]. Both estimators were run 10 000 times, each time with N new random vectors. Fig. 2 shows histograms of the resulting estimates of the standard deviation, together with the exact value. It can be observed that (12) in this example is more than an order of magnitude more accurate than the jackknife with $g = 100$.

The accuracy of the jackknife improves with increasing g, and it is most accurate when each group consists of a single sample, i.e., when $g = N$. In this extreme case, the jackknife simplifies into (15)—but this is not how the jackknife was applied in previous studies [17]–[19].

Having established the usefulness of the new variance estimator, we proceed to estimate the quantizer constant G of Λ_{16} with high accuracy. Numerically evaluating (12) and (16) for the mean and (15) and (17) for the standard deviation, using $N = 4 \cdot 10^{12}$ random 16-dimensional vectors, we obtain

$$\hat{G} = 0.068297616\,, \tag{18}$$

$$\sqrt{\widehat{\operatorname{var}}\,\hat{G}} = 0.000000009\,. \tag{19}$$

The difference between $\hat{G}$ and the exact G in (5) is only 0.7 standard deviations, which may serve as a numerical verification of the face hierarchy. The results are also in agreement with the previous (less accurate) estimate of the same constant in [17, Eq. (13)].

V. The Algorithm

Our algorithm[2] is described in detail in [15], which builds on previous methods for finding all relevant vectors [23] and faces [24]. In this section, we briefly summarize the main concept and present minor modifications to the methods of [15].

The basic approach remains the same: We first find all relevant vectors, i.e., normals of the facets, and all the vertices of the Voronoi region. The hierarchy of subfaces of the facets is then built by recursively intersecting the sets of vertices of parent faces. The computational cost is kept low by finding the classes of faces equivalent under $\operatorname{Aut}(\Lambda_{16})$ and then only constructing the child faces of one (arbitrarily chosen) representative face per class. In total, only 159 143 faces are constructed explicitly.

The classification of faces is performed iteratively as described in [15, Section 2.4.4]. In this method, we begin identifying equivalent faces using a proper subgroup $\mathcal{U} \subset \operatorname{Aut}(\Lambda_{16})$, which creates classes of faces under $\mathcal{U}$. The set consisting of one (arbitrary) representative per class is then classified using another subgroup $\mathcal{U}'$. This can be repeated with different subgroups until we finally use the full group $\operatorname{Aut}(\Lambda_{16})$. For Λ_{16}, we found that a good option is to use only a single subgroup $\mathcal{U}$, chosen as the stabilizer of the relevant vector $\boldsymbol{n}_2$ with a stabilizer size of 1 451 520 (see Tab. I).

We made three changes to the method in [15], which affect how the equivalence of two faces is tested and how the orbits and stabilizers of individual vectors are constructed. We now describe these changes in turn, briefly revisiting the respective previous methods followed by our new algorithms.

A. Testing the equivalence of faces

Our previous method of testing whether a face F is equivalent to another face F' under a group $\mathcal{G}$ is based on the following idea.[3] For each face, we take a set of vectors that uniquely identifies that face. We use either the set of relevant vectors associated with the facets containing the face (i.e., the "normal vectors" of the face) or alternatively the face's vertices. The choice depends on the number of vectors in either of the two sets and on their classification under $\mathcal{G}$. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ be the vectors of F and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ be those of F'. We order these vectors such that $\boldsymbol{x}_i$ is equivalent to $\boldsymbol{y}_i$ for all i (if that is not possible, the faces are inequivalent). We then form the sets of all transformations between pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for all i. If the intersection of these sets is non-empty, it consists of transformations taking F into F'. If it is empty, however, we permute one of the sets and try again. The faces

[2] The algorithms are implemented in *Python* and the data types "List" and "Dictionary" we use in the code listings are meant to behave like the respective Python types. Group-theoretic aspects make use of *GAP* [20], [21], which is called from Python using *gappy* [22].

[3] Here, $\mathcal{G}$ is either $\operatorname{Aut}(\Lambda_{16})$ or $\mathcal{U}$.

are inequivalent if and only if all permutations lead to empty intersections of the sets of transformations.

In principle, the full set of transformations between any two equivalent vectors can easily be constructed as follows. Let $\boldsymbol{x} = g_x \boldsymbol{x}^{\text{rep}}$ and $\boldsymbol{y} = g_y \boldsymbol{x}^{\text{rep}}$ be two equivalent vectors with $g_x, g_y \in \mathcal{G}$ and $\boldsymbol{x}^{\text{rep}}$ representing their equivalence class. Then, the full set of transformations in $\mathcal{G}$ taking $\boldsymbol{x}$ into $\boldsymbol{y}$ is [15]

$$\mathcal{T}_{xy} = g_y \,\text{Stab}_{\mathcal{G}}(\boldsymbol{x}^{\text{rep}}) g_x^{-1} \,, \tag{20}$$

where $\text{Stab}_{\mathcal{G}}(\boldsymbol{x}^{\text{rep}})$ is the stabilizer of $\boldsymbol{x}^{\text{rep}}$ in $\mathcal{G}$.

From Tab. I, we see that for Λ_{16}, the sets (20) contain between $1\,344$ and $20\,643\,840$ elements. When forming the intersections using *GAP*, these sets are held in memory, which becomes a problem when multiple intersections need to be calculated.

We now describe a memory-efficient alternative, shown in Alg. 1. As in [15], this method is used after ensuring that F and F' have the same number of vertices and number of normal vectors, and that the respective sets of vectors can be ordered such that $\boldsymbol{x}_i \sim \boldsymbol{y}_i$ for all i.

The main idea is to fix one vector $\boldsymbol{x}$ of F and then construct all transformations

$$\mathcal{T}_x = \bigcup_{\boldsymbol{y} \in \mathcal{Y}} \mathcal{T}_{xy} \tag{21}$$

taking $\boldsymbol{x}$ into any of the vectors $\boldsymbol{y} \in \mathcal{Y}$, where $\mathcal{Y}$ denotes the vectors of F'. Clearly, if F and F' are equivalent, say $gF = F'$ for some $g \in \mathcal{G}$, then g takes $\boldsymbol{x}$ into one of the vectors $\boldsymbol{y}$ of F' and thus $g \in \mathcal{T}_x$. Choosing $\boldsymbol{x}$ as the vector with the smallest stabilizer and fewest equivalent vectors of F', $\mathcal{T}_x$ will often be very small and can be checked one by one. However, even if the smallest stabilizer is large, the elements of $\mathcal{T}_x$ can be enumerated without holding the full set in memory.

Alg. 1 performs this test as follows. In lines 6 and 7, $\boldsymbol{x}$ is chosen as the vector with the smallest stabilizer and, if there are multiple possibilities, then the one with the smallest number of equivalent vectors of F'. In line 10, we store the set of these equivalent vectors as $\mathcal{Y}_x$. Independently from the choice of $\boldsymbol{x}$, let $\mathcal{D}$ be the smaller of the sets of vertices and of normal vectors of F (lines 12–17). We choose $\mathcal{D}'$ analogously for F'. Since the stabilizer is a group, we can use methods in *GAP* to iterate over all its elements in line 18, while holding only one element in memory at any given time. For each element $g_s \in \text{Stab}_{\mathcal{G}}(\boldsymbol{x}^{\text{rep}})$ and each $\boldsymbol{y} \in \mathcal{Y}_x$, we form the transformation (line 21)

$$g = g_y g_s g_x^{-1} \tag{22}$$

and evaluate if the two sets $g\mathcal{D}$ and $\mathcal{D}'$ are equal. If they are, then F is equivalent to F' and $gF = F'$. If they are unequal for all $g_s \in \text{Stab}_{\mathcal{G}}(\boldsymbol{x}^{\text{rep}})$ and all $\boldsymbol{y} \in \mathcal{Y}_x$, then the two faces are inequivalent under $\mathcal{G}$.

B. Constructing the orbit of a vector

We use a variation of the standard orbit enumeration technique as implemented, e.g., in [20]. Alg. 2 constructs the orbit of a vector $\boldsymbol{x}$ under a group $\mathcal{G}$ and stores the group elements taking $\boldsymbol{x}$ to the elements in its orbit. These group elements are needed in the procedure TRANSFORMOF in Alg. 1. The result is stored as a dictionary, where each key–value pair consists of an element $\boldsymbol{y}$ of the orbit as key and one arbitrary transformation matrix taking $\boldsymbol{x}$ into $\boldsymbol{y}$ as value. We will call such a dictionary an *orbit map*.

For vertices, most of the group elements and, in fact, most of the vertices themselves are not needed in Alg. 1. Since child faces are constructed only for the fixed representative parent faces, only the vertices of the representative facets can appear. Our orbit algorithm therefore selectively stores only some of the group elements, which is decided in Alg. 2 using a *condition* function. This significantly reduces the memory usage for the large orbits of vertices. For Λ_{16}, only the group elements corresponding to $1\,067\,070$ out of all $201\,343\,200$

Algorithm 1 Evaluate if two faces F and F' are equivalent under $\mathcal{G}$. If they are, return a transformation $g \in \mathcal{G}$ taking F into F', otherwise return NULL. We define $|\mathcal{X}|$ as the number of elements in a set $\mathcal{X}$ and $\arg\min_{\boldsymbol{x} \in \mathcal{X}} f$ as the function returning the subset of $\mathcal{X}$ for which $f(\boldsymbol{x})$ is smallest.

1: **procedure** FINDTRANSFORMATION(F, F', $\mathcal{G}$)
2: $\quad\mathcal{V} \leftarrow$ vertices of F
3: $\quad\mathcal{V}' \leftarrow$ vertices of F'
4: $\quad\mathcal{N} \leftarrow$ normal vectors of F
5: $\quad\mathcal{N}' \leftarrow$ normal vectors of F'
$\quad$There may be multiple minima, which are all captured:
6: $\quad\mathcal{M} \leftarrow \arg\min_{\boldsymbol{x} \in \mathcal{V} \cup \mathcal{N}} |\text{Stab}_{\mathcal{G}}(\text{REPOF}(\boldsymbol{x}, \mathcal{G}))|$
$\quad$Any of the elements with the smallest number of equivalent vectors of F' is selected:
7: $\quad\boldsymbol{x} \leftarrow \text{ANY}(\arg\min_{\boldsymbol{x} \in \mathcal{M}} |\{\boldsymbol{y} \in \mathcal{V}' \cup \mathcal{N}' : \boldsymbol{y} \sim \boldsymbol{x}\}|)$
8: $\quad\boldsymbol{x}^{\text{rep}} \leftarrow \text{REPOF}(\boldsymbol{x}, \mathcal{G})$
9: $\quad g_x^{-1} \leftarrow$ inverse of $\text{TRANSFORMOF}(\boldsymbol{x}, \mathcal{G})$
10: $\quad\mathcal{Y}_x \leftarrow \{\boldsymbol{y} \in \mathcal{V}' \cup \mathcal{N}' : \boldsymbol{y} \sim \boldsymbol{x}\}$
11: $\quad\mathcal{T}_y \leftarrow \{\text{TRANSFORMOF}(\boldsymbol{y}, \mathcal{G}) : \boldsymbol{y} \in \mathcal{Y}_x\}$
12: $\quad$**if** $|\mathcal{V}| < |\mathcal{N}|$ **then**
13: $\quad\quad\mathcal{D} \leftarrow \mathcal{V}$
14: $\quad\quad\mathcal{D}' \leftarrow \mathcal{V}'$
15: $\quad$**else**
16: $\quad\quad\mathcal{D} \leftarrow \mathcal{N}$
17: $\quad\quad\mathcal{D}' \leftarrow \mathcal{N}'$
18: $\quad$**for all** $g_s \in \text{Stab}_{\mathcal{G}}(\boldsymbol{x}^{\text{rep}})$ **do**
19: $\quad\quad\bar{g} \leftarrow g_s g_x^{-1}$
20: $\quad\quad$**for all** $g_y \in \mathcal{T}_y$ **do**
21: $\quad\quad\quad g \leftarrow g_y \bar{g}$
22: $\quad\quad\quad$**if** $g\mathcal{D} = \mathcal{D}'$ **then**
23: $\quad\quad\quad\quad$**return** g
24: $\quad$**return** NULL

Utility functions:

- ANY($\mathcal{X}$) returns an arbitrary element of $\mathcal{X}$
- REPOF($\boldsymbol{x}$, $\mathcal{G}$) returns a representative of $\boldsymbol{x}$, assuming that all vectors have been classified under $\mathcal{G}$ and an arbitrary but fixed choice of class representatives has been made
- TRANSFORMOF($\boldsymbol{x}$, $\mathcal{G}$) returns $g_x \in \mathcal{G}$ such that $\boldsymbol{x} = g_x \text{REPOF}(\boldsymbol{x}, \mathcal{G})$, again assuming that vectors have been classified under $\mathcal{G}$ and that (at least) one group element taking its representative into $\boldsymbol{x}$ is known

Algorithm 2 Construct the orbit of a vector $\boldsymbol{x}$ under a group $\mathcal{G}$. The group is given as a set *gens* of transformation matrices generating the full group. For Λ_{16} we use $gens = \{\boldsymbol{M}_1, \boldsymbol{M}_2\}$, where $\boldsymbol{M}_1, \boldsymbol{M}_2$ are given in Sec. III. The *condition* is a boolean function of one vector and specifies if the transformation matrix should be stored for the given vector. This procedure returns a set of all vectors in the orbit as well as an orbit map. See the main text for details.

```
1:  procedure ORBIT(x, gens, condition)
2:      orbit ← {x}
3:      orbit_map ← new empty Dictionary
4:      orbit_map[x] ← identity matrix
5:      pool ← copy of orbit_map
6:      while pool is not empty do
7:          new_pool ← new empty Dictionary
8:          for all y ∈ keys of pool do
9:              h ← pool[y]
10:             for all g ∈ gens do
11:                 y' ← gy
12:                 if y' ∉ orbit then
13:                     orbit ← orbit ∪ {y'}
14:                     h' ← gh
15:                     new_pool[y'] ← h'
                        See the main text for this if-statement:
16:                     if condition(y') then
17:                         orbit_map[y'] ← h'
18:         pool ← new_pool
19:     return orbit, orbit_map
```

vertices are needed.[4]

The idea of the standard orbit algorithm is to repeatedly apply the generators of the group to the initial and the newly constructed vectors until no new vector appears. This is used in Alg. 2, where the *pool* and *new_pool* variables keep track of which new vectors have appeared in the last iteration. In lines 16–17, we conditionally store the vector and its transformation in *orbit_map*. If all vectors are known, the new pool remains empty and the termination condition of the while-loop is satisfied. When constructing the orbits of vertices, the *condition* is chosen to evaluate to *true* only when the vector lies in one of the representative facets. For relevant vectors, *condition* is set to always evaluate to *true*.

C. Constructing the stabilizer of a vector

The third change to the method in [15] is an algorithm to construct the stabilizer of a vector under a group $\mathcal{G}$. Our method is again inspired by a standard orbit-stabilizer algorithm such as the one implemented in [20]. Stabilizers are needed in line 18 of Alg. 1, where we iterate over all elements of the stabilizer of one of the representative vectors. For $\mathcal{G} = \mathrm{Aut}(\Lambda_{16})$, there are in total 8 representative vectors listed in Tab. I. We previously let *GAP* find the stabilizer of a vector. With the knowledge about each vector's orbit size, however, we can implement a more efficient method.

[4]Because some vertices appear in both facets, this number will vary depending on which facets are chosen as representatives.

Algorithm 3 Construct the stabilizer of a vector $\boldsymbol{x}$ in $\mathcal{G}$ whose orbit size is known. As in Alg. 2, the group $\mathcal{G}$ is given as a set *gens* of generator matrices. See the main text for details.

```
1:  procedure STABILIZER(x, gens, orbit_size)
2:      G ← GAP group from gens
3:      stab_size ← |G|/orbit_size
4:      stab_gens ← new empty List
5:      stab ← GAP group containing only I_16
6:      orbit_map ← new empty Dictionary
7:      orbit_map[x] ← identity matrix
8:      for all g ∈ G do
9:          x' ← gx
10:         if x' ∈ keys of orbit_map then
11:             g' ← orbit_map[x']
12:             g_s ← g^{-1} g'
13:             if g_s ∉ stab then
14:                 append g_s to stab_gens
15:                 stab ← GAP group from stab_gens
16:                 if |stab| = stab_size then
17:                     return stab
18:         else
19:             orbit_map[x] ← g
```

In Alg. 3, we construct elements of the orbit by applying different group elements to the vector $\boldsymbol{x}$ (line 9). Any vector $\boldsymbol{x}'$ that is visited this way is stored together with the corresponding group element in an orbit map (line 19). Whenever we encounter a vector $\boldsymbol{x}'$ previously found, we retrieve the stored group element g' (line 11). Since $g\boldsymbol{x} = g'\boldsymbol{x}$, we have $g^{-1}g'\boldsymbol{x} = \boldsymbol{x}$ and so $g_s = g^{-1}g'$ is an element of the stabilizer of $\boldsymbol{x}$. If it is not yet an element of the subgroup $stab \subseteq \mathrm{Stab}_{\mathcal{G}}(\boldsymbol{x})$ found thus far, it is added to the list of group generators in line 14. After updating *stab* in line 15, we check if it is complete by comparing its size against the known stabilizer size.

This is made efficient by two facts. First, due to the "birthday paradox" [25, Section 3], the first coincidence in line 10 occurs on average after $1 + \sum_{n=1}^{N} \prod_{i=1}^{n-1}(1 - i/N)$ group elements (see the second unnumbered equation below [25, Eq. (12)]), where N is the size of the orbit of $\boldsymbol{x}$ under $\mathcal{G}$. For $\mathrm{Aut}(\Lambda_{16})$, this means that the first element of the stabilizers of the vectors in Tab. I is found after about 83 (for $\boldsymbol{n}_1$ and $\boldsymbol{v}_1$) to 10 210 (for $\boldsymbol{v}_2, \boldsymbol{v}_4, \boldsymbol{v}_5$) iterations. Second, the stabilizers are often generated by very few group elements. In the case of Λ_{16}, the set of all 8 stabilizers is found within minutes on a single core, since each stabilizer can be generated by only two generators.

VI. Conclusions

In this work, we provide a complete account of the relevant vectors, vertices, and face classes of the Voronoi region of the Barnes–Wall lattice Λ_{16}. This is used to calculate the exact second moment of Λ_{16}. In order to obtain these results, we improve our algorithm [15], allowing it to be used with larger symmetry groups than previously possible. We believe that our algorithm can be used to analyse the Voronoi regions of

many lattices with known symmetry group, potentially even in dimensions higher than 16.

Using Monte-Carlo integration, the exact value of the second moment is numerically verified. Furthermore, it is shown that the variance of the numerical result can be approximated with much higher accuracy than conventionally obtained with the jackknife estimator. This may provide significant improvements in numerical second moment estimates in the future.

REFERENCES

[1] E. S. Barnes and G. E. Wall, "Some extreme forms defined in terms of Abelian groups," *Journal of the Australian Mathematical Society*, vol. 1, no. 1, pp. 47–63, Aug. 1959.

[2] G. D. Forney, Jr., "Coset codes—part I: Introduction and geometrical classification," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1123–1151, Sept. 1988.

[3] ——, "Coset codes—part II: Binary lattices and related codes," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1152–1187, Sept. 1988.

[4] A. J. Hahn, "The coset lattices of E. S. Barnes and G. E. Wall," *Journal of the Australian Mathematical Society. Series A*, vol. 49, no. 3, pp. 418–433, Dec. 1990.

[5] G. Nebe, E. M. Rains, and N. J. A. Sloane, "A simple construction for the barnes-wall lattices," in *Codes, Graphs, and Systems: A Celebration of the Life and Career of G. David Forney, Jr. on the Occasion of his Sixtieth Birthday*, R. E. Blahut and R. Koetter, Eds. Boston, MA: Springer US, 2002, pp. 333–342. [Online]. Available: https://doi.org/10.1007/978-1-4615-0895-3_19

[6] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, 3rd ed. New York, NY: Springer, 1999. [Online]. Available: https://doi.org/10.1007/978-1-4757-6568-7

[7] M. S. Viazovska, "The sphere packing problem in dimension 8," *Annals of Mathematics*, pp. 991–1015, 2017. [Online]. Available: https://doi.org/10.4007/annals.2017.185.3.7

[8] J.-P. Adoul, "Lattice and trellis coded quantizations for efficient coding of speech," in *Speech Recognition and Coding*, A. J. R. Ayuso and J. M. L. Soler, Eds. Berlin, Germany: Springer, 1995, ch. 57, pp. 405–422.

[9] S. Lyu, L. Liu, J. Lai, C. Ling, and H. Chen, "Lattice codes for lattice-based PKE," 2022. [Online]. Available: https://arxiv.org/abs/2208.13325

[10] A. R. Calderbank, E. M. Rains, P. Shor, and N. J. A. Sloane, "Quantum error correction via codes over GF(4)," *IEEE Trans. Inf. Theory*, vol. 44, no. 4, pp. 1369–1387, 1998.

[11] P. Buser and P. Sarnak, "On the period matrix of a Riemann surface of large genus (with an Appendix by J.H. Conway and N.J.A. Sloane)," *Inventiones mathematicae*, vol. 117, no. 1, pp. 27–56, 1994.

[12] J. H. Conway and N. J. A. Sloane, "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 211–226, Mar. 1982. [Online]. Available: https://doi.org/10.1109/TIT.1982.1056483

[13] J. Mueller, M. Neunhöffer, F. Noeske, and M. Horn, "orb, Methods to enumerate orbits, Version 4.8.3," Sept. 2019, GAP package. [Online]. Available: https://gap-packages.github.io/orb

[14] The arXiv version of this paper, available at https://arxiv.org/abs/2301.06092, has an ancillary file containing a catalog of the face classes of Λ_{16}. It can be accessed at https://arxiv.org/src/2301.06092v1/anc.

[15] D. Pook-Kolb, B. Allen, and E. Agrell, "Exact calculation of quantizer constants for arbitrary lattices," 2022. [Online]. Available: https://arxiv.org/abs/2211.01987

[16] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. Elsevier, 1977, vol. 16.

[17] J. H. Conway and N. J. A. Sloane, "On the Voronoi regions of certain lattices," *SIAM J. Alg. Disc. Meth.*, vol. 5, no. 3, pp. 294–305, Sept. 1984. [Online]. Available: https://doi.org/10.1137/0605031

[18] E. Agrell and T. Eriksson, "Optimization of lattices for quantization," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 1814–1828, Sept. 1998.

[19] S. Lyu, Z. Wang, C. Ling, and H. Chen, "Better lattice quantizers constructed from complex integers," 2022. [Online]. Available: https://arxiv.org/abs/2204.01105

[20] The GAP Group, "GAP – Groups, Algorithms, and Programming, Version 4.12dev." [Online]. Available: https://www.gap-system.org

[21] ——, "Main development repository for GAP – Groups, Algorithms, Programming," Oct. 2021, Commit 401c797476b787e748a3890be4ce95ae4e5d52ae. [Online]. Available: https://github.com/gap-system/gap

[22] E. M. Bray, "gappy – a Python interface to GAP, Version 0.1.0a4." [Online]. Available: https://github.com/embray/gappy

[23] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, 2002.

[24] B. Allen and E. Agrell, "The optimal lattice quantizer in nine dimensions," *Annalen der Physik*, vol. 533, no. 12, p. 2100259, Dec. 2021. [Online]. Available: https://doi.org/10.1002/andp.202100259

[25] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, 1992.