Generalizing CGAL Periodic Delaunay Triangulations

Georg Osang
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria
georg.osang@ist.ac.at

Mael Rouxel-Labbé
GeometryFactory, Grasse, France
mael.rouxel.labbe@geometryfactory.com

Monique Teillaud
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
Monique.Teillaud@inria.fr

Abstract

Even though Delaunay originally introduced his famous triangulations in the case of infinite point sets with translational periodicity, a software that computes such triangulations in the general case is not yet available, to the best of our knowledge. Combining and generalizing previous work, we present a practical algorithm for computing such triangulations. The algorithm has been implemented and experiments show that its performance is as good as the one of the CGAL package, which is restricted to cubic periodicity.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Delaunay triangulation, lattice, algorithm, software, experiments

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.75

Supplementary Material `https://members.loria.fr/Monique.Teillaud/CGAL_periodicDT_ESA20/`

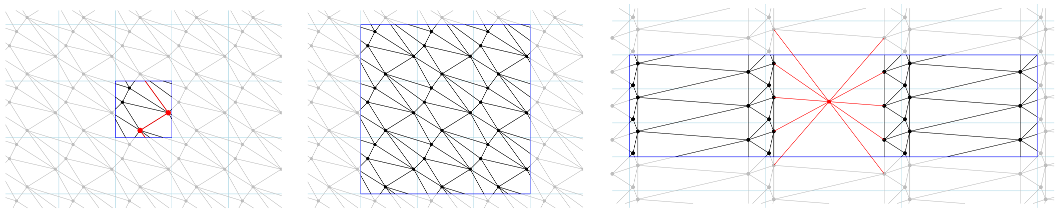
Funding *Georg Osang*: This author is partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant no. 788183.
Monique Teillaud: This author is partially supported by the grants ANR-17-CE40-0033 of the French National Research Agency ANR (project SoS) and INTER/ANR/16/11554412/SoS of the Luxembourg National Research fund FNR (`https://members.loria.fr/Monique.Teillaud/collab/SoS/`).

1 Introduction

Delaunay triangulations are one of the most prominent structures in computational geometry. While nowadays many applications use Delaunay triangulations of finite point sets in Euclidean space, Delaunay originally introduced the notion in the context of infinite point sets with translational periodicity [9]. Such periodic point sets are abundant in fields such as crystallography and material sciences; thus their communities would benefit from software that computes Delaunay triangulations of infinite periodic point sets in Euclidean space. Specifically, given a d-dimensional lattice and its associated translation group, the orbits of a given finite point set in $\mathbb{R}^d$ with respect to this translation group define a periodic point set. Our aim is to compute a finite representation of the periodic Delaunay triangulation of such a periodic point set, specifically a projection of the triangulation onto the flat d-torus that is the quotient space of $\mathbb{R}^d$ under the action of the translation group.

The first algorithm for this problem was already outlined in 1997 [12], yet a robust and efficient implementation for this problem does not exist to date as far as we know. The Voro++ library [18] is focused on crystallographic applications in 3 dimensions; however it is limited to orthogonal lattices. Zeo++ [22] extends its functionality to arbitrary 3-dimensional lattices. Both libraries compute the Voronoi cell of a given input point as an intersection of half-spaces, by searching for other points around it that have an influence on its Voronoi cell. The combinatorics of the Delaunay triangulation cannot be easily accessed. The CGAL library [17] provides packages for periodic Euclidean Delaunay triangulations in 2D and 3D, which currently are limited to the integer lattice, referred to as the square and cubic setting and 2 and 3 dimensions, respectively [15, 6, 5]. We propose an addition to CGAL that extends this functionality to arbitrary lattices.

The algorithm by Dolbilin and Huson [12] creates 3^d copies of each input point and computes their finite Delaunay triangulation, from which a representation of the periodic Delaunay triangulation is extracted. The CGAL algorithm [4, 7] computes the triangulation in a finitely-sheeted covering space of the d-torus. It is based on the classical incremental algorithm by Bowyer and Watson [3, 21], and requires that the triangulation be a simplicial complex at any given time. Let us quickly recall that a triangulation is a simplicial complex, or is *simplicial* for short, if each of its simplices consists of a set of distinct vertices, and the intersection of any two simplices is either empty or a simplex. Operating directly on the d-torus does not guarantee this, see Figure 1a. Thus, in the cubic setting, a 3^d-sheeted cover is used (Figure 1b) until sufficiently many points have been inserted to guarantee that the triangulation in the 1-sheeted cover is a simplicial complex. Unfortunately, a 3^d-sheeted cover is not sufficient for more general periodic point sets: As the 3^d copies of each point have to be inserted iteratively into the 3^d-sheeted cover, simpliciality can be violated in the intermediate stages of point insertion, see Figure 1c. While there always exists a finitely-sheeted covering space [7] that ensures simpliciality, the number of sheets might be prohibitively large. Thus, we propose a different approach.



(a) The intersection of the two red edges is not a simplex but a set of two vertices. Thus the triangulation is not simplicial.

(b) The 9-sheeted cover guarantees a simplicial triangulation in the square setting.

(c) In the non-square setting, incremental point insertion of the 9 copies into the 9-sheeted cover can violate simpliciality. Here, the first of 9 copies is being inserted.

Figure 1 Representation of the projections of the periodic Delaunay triangulation into the 1-sheeted (left) and 9-sheeted cover (middle, right) of the 2-torus.

Overview. After formally defining the problem in Section 2, we propose an algorithm (Section 3) for periodic Delaunay triangulations that combines two different approaches and consists of two phases, both of which use Bowyer-Watson's algorithm. While for 2-dimensional spaces algorithms based on flips circumvent the simpliciality requirement [10], we stick to Bowyer-Watson's algorithm as it easily generalizes to 3 (and higher) dimensions. Furthermore it enables an efficient, clean, and easily maintainable implementation. The first phase of our algorithm (Section 3.2) refines the algorithm by Dolbilin and Huson [12], and its implementation details are based on some new results. It uses 3^d copies of each input point, regardless of the lattice, and computes a finite Euclidean Delaunay triangulation on this point set, from which a representation of the Delaunay triangulation on the d-torus is obtained. Once a simpliciality criterion is met, our algorithm switches to the second phase

(Section 3.4), which conceptually follows the CGAL implementation of the cubic case [4]. It operates directly on the d-torus, maintaining only one copy of each input point, and thus provides better insertion running times than phase 1. A first version of our open-source implementation in 2D and 3D is publicly available.[1] This implementation is expected to be an integral part of CGAL in a near future release. Experiments (Section 4) show similar performances as the CGAL implementation restricted to cubic lattices [5]. We close with a discussion of future extensions in Section 5.

2 Preliminaries

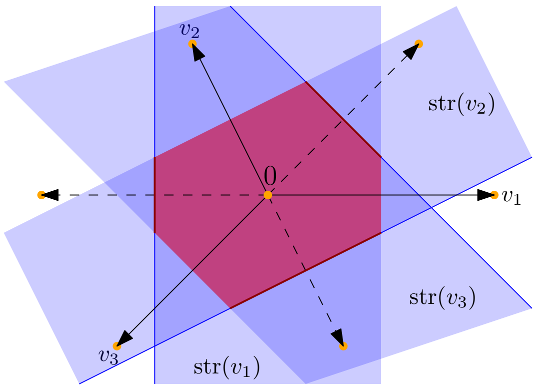
Let us recall various notions [13, 8] that are employed throughout the algorithm. Let $B = \{b_1, b_2, \ldots, b_d\}$ be a basis of $\mathbb{R}^d$. The point set $\Lambda := \{\sum_{i=1}^{d} z_i b_i \colon z \in \mathbb{Z}^d\}$ is called a *lattice*, and B is its *lattice basis*. The lattice Λ is associated with the translation group Γ consisting of the translations $\phi_\lambda \colon \mathbb{R}^d \to \mathbb{R}^d$ mapping the origin to λ, for each $\lambda \in \Lambda$. The group Γ acts on $\mathbb{R}^d$ and each translation of Γ maps Λ onto itself. We denote the length of the shortest non-zero lattice vector as $\mathrm{sv}(\Lambda)$. For a given lattice basis B, we call $B_{\mathrm{sup}} := B \cup \{b_0\}$ with $b_0 = -\sum_{i=1}^{d} b_i$ its *superbase*. A superbase is *obtuse* if for any pair b_i and b_j, $\langle b_i, b_j \rangle \leq 0$. A basis is *reduced* if its superbase is obtuse [13, Definition 4.4]. This notion is defined in such a way that we can easily compute $\mathrm{sv}(\Lambda)$ and Dirichlet domains. The *Dirichlet domain* of a lattice point $\lambda \in \Lambda$ is the region of λ in the Voronoi tesselation of Λ, or more formally $\mathrm{dom}(\lambda, \Lambda) := \{p \in \mathbb{R}^d \colon \|p - \lambda\| \leq \|p - \nu\| \; \forall \nu \in \Lambda\}$. It is a convex polytope, and we call the lattice Λ *generic* if each vertex of $\mathrm{dom}(0, \Lambda)$ is incident to the Dirichlet domains of exactly d other lattice points. For 2-dimensional generic lattices the Dirichlet domains are hexagons, for 3-dimensional generic lattices they are combinatorially equivalent to truncated cubes.

For a lattice vector λ, $\mathrm{str}(\lambda) = \{p \in \mathbb{R}^d \colon -0.5 \leq \frac{\langle p, \lambda \rangle}{\langle \lambda, \lambda \rangle} < 0.5\}$ is an infinite half-open strip that contains the subspace orthogonal to λ through the origin. Then $\mathrm{dom}(0, \Lambda)$ is the closure of the intersection of these strips for all non-zero lattice vectors. However, as $\mathrm{dom}(0, \Lambda)$ only has a finite number of facets, there must be a finite subset of strips whose closed intersection yields $\mathrm{dom}(0, \Lambda)$. Let $\mathcal{V}$ be the minimal set of lattice vectors (together with their negates) such that the closure of $\bigcap_{v \in \mathcal{V}} \mathrm{str}(v)$ is $\mathrm{dom}(0, \Lambda)$. The vectors in $\mathcal{V}$ are commonly called *Voronoi-relevant vectors*. Each of them is a normal vector of a facet of the Dirichlet domain of 0, and thus there are at most $2(2^d - 1)$ Voronoi relevant vectors [13, Theorem 3.6]. Let $\mathcal{V}^+ \sqcup \mathcal{V}^-$ be a partition of $\mathcal{V}$ such that if $v \in \mathcal{V}^+$, then $-v \in \mathcal{V}^-$, and vice versa. For a fixed choice of $\mathcal{V}^+$ (and implicitly $\mathcal{V}^-$), we define the *canonical* domain $\mathrm{dom}^{◁}(0, \Lambda) := \bigcap_{v \in \mathcal{V}^+} \mathrm{str}(v)$ (Figure 2). Its closure is $\mathrm{dom}(0, \Lambda)$, and its images under Γ, denoted $\mathrm{dom}^{◁}(\lambda, \Lambda) := \phi_\lambda(\mathrm{dom}^{◁}(0, \Lambda))$ for $\lambda \in \Lambda$, form a partition of $\mathbb{R}^d$. With $k\Lambda$ for $k \in \mathbb{Z}$ referring to the lattice with basis $\{kb_1, \ldots, kb_d\}$, we note that $\mathrm{dom}^{◁}(0, k\Lambda)$ is $\mathrm{dom}^{◁}(0, \Lambda)$ scaled by a factor of k.

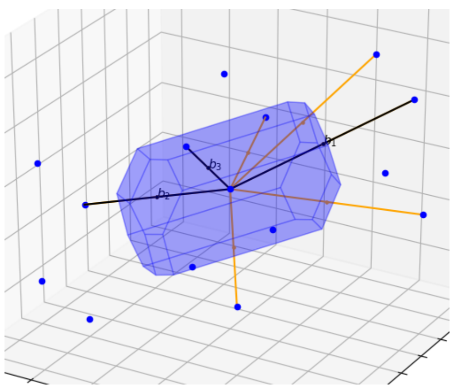
For $d \leq 3$, if we have a reduced lattice basis B with its superbase B_{sup}, then $\mathcal{V}$ is a subset of $\{\sum_{v \in S} v \colon S \subset B_{\mathrm{sup}}, S \neq \emptyset, S \neq B_{\mathrm{sup}}\}$ [8, Theorems 3 and 8], with equality if the lattice is generic. Note that $\mathrm{sv}(\Lambda)$ can be obtained as the length of the shortest vector in $\mathcal{V}$.

Delaunay triangulations. The *Delaunay triangulation* $\mathrm{Del}(X)$ of an input point set $X \subset \mathbb{R}^d$ is a collection of simplices up to dimension d whose vertex set is X and each d-simplex (which we refer to as a *cell*) corresponds to a set of $d+1$ points whose open circumscribing ball does not contain any other points of X. We call the $(d-1)$-simplices of $\mathrm{Del}(X)$ its *facets*.

[1] `https://members.loria.fr/Monique.Teillaud/CGAL_periodicDT_ESA20/`



(a) $\mathrm{dom}^{◁}(0, \Lambda)$ (red) is the intersection of the half-open strips (blue) of $\mathcal{V}^+ = \{v_1, v_2, v_3\}$.



(b) $\mathrm{dom}^{◁}(0, \Lambda)$ of a 3-dimensional lattice (blue). Lattice basis B in black, $\mathcal{V}^+ \supset B$ in orange.

Figure 2 Canonical domains and Voronoi relevant vectors for lattices in 2D and 3D.

Given a lattice Λ and a finite set of points X, we get the *periodic point set* $\Gamma X := \{\phi_\lambda(x) : x \in X \text{ and } \lambda \in \Lambda\}$ consisting of the elements of the orbits of X under Γ. ΓX is globally invariant under Γ. Then $\mathrm{Del}(\Gamma X)$ is the *periodic Delaunay triangulation* of the infinite point set ΓX. Note that we can ignore degeneracies in ΓX by using the symbolic perturbation provided by CGAL [11]: it is translation-invariant, so, degeneracies are triangulated in a consistent way, which ensures that the computed $\mathrm{Del}(\Gamma X)$ is actually invariant under Γ. The orbit space $\mathbb{R}^d/\Gamma$ is a *flat torus*, and we denote its projection map as $\pi\colon \mathbb{R}^d \to \mathbb{R}^d/\Gamma$. The *torus triangulation* $\mathrm{Del}(\Gamma X)/\Gamma$ is the projection of $\mathrm{Del}(\Gamma X)$ into $\mathbb{R}^d/\Gamma$. Using $\mathrm{dom}^{◁}(0, \Lambda)$ as a geometric representation of the torus, we can use $X_0 := \Gamma X \cap \mathrm{dom}^{◁}(0, \Lambda)$ as *canonical* representatives of the vertex set of $\mathrm{Del}(\Gamma X)/\Gamma$. While $\mathrm{Del}(\Gamma X)/\Gamma$ gives us a finite representation of $\mathrm{Del}(\Gamma X)$, unlike $\mathrm{Del}(\Gamma X)$ it is not necessarily simplicial (see Figure 1a).

3 Algorithm

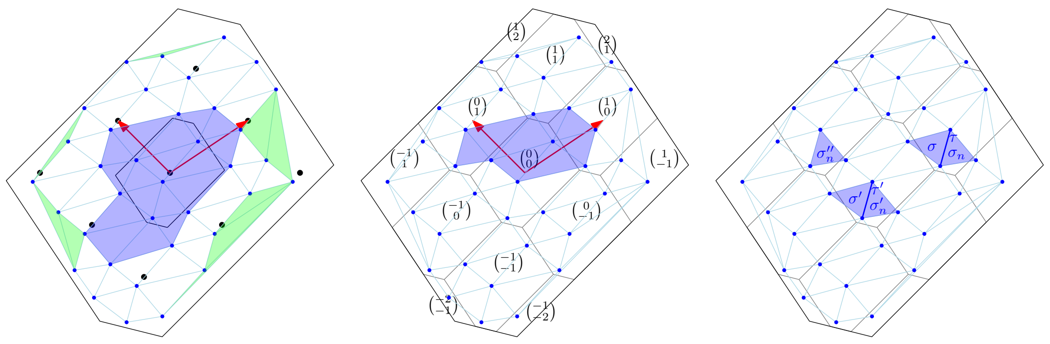
The input to our algorithm is a lattice basis B' for Λ and a set of points X defining the periodic point set. The output is an object representing $\mathrm{Del}(\Gamma X)/\Gamma$. This object provides a uniform interface that, regardless of the internal state of our algorithm, allows the user to access the properties of Λ as well as the torus triangulation $\mathrm{Del}(\Gamma X)/\Gamma$. For Λ this includes the reduced basis $B = \{b_1, \dots, b_d\}$. For $\mathrm{Del}(\Gamma X)/\Gamma$ this includes the set X_0 of canonical representatives for its vertex set, and its cells. Cells are not solely defined by their vertex set, but have additional geometric information attached. Specifically, each vertex of a cell is represented as a point x from X_0 with an associated offset $o = (o_1, \dots, o_d)$, which is an integer vector. The geometric location of the vertex then is $x + \sum_{i=1}^{d} o_i b_i$. In alignment with other CGAL triangulations, we also provide access to simplices of lower dimensions, represented with associated vertex offsets akin to cells, as well as neighborhood relations. These include querying a cell for its adjacent cells, or querying a vertex for its incident cells or lower dimensional simplices. While many steps generalize, we restrict our focus to 2- and 3-dimensional triangulations, which are the most widely used.

Internally, our algorithm operates in two phases, which use two different data structures, respectively: The first phase maintains a finite Euclidean Delaunay triangulation while the second phase maintains a triangulation of $\mathrm{Del}(\Gamma X)/\Gamma$.

Let us now recall the result that is crucial to the first phase of our algorithm. For a point set X, let $X_3 := \mathrm{dom}^{\LEFTcircle}(0, 3\Lambda) \cap \Gamma X$, i.e. all periodic copies of X that lie within the Dirichlet domain of 0 scaled by a factor of 3. We call a cell of $\mathrm{Del}(X_3)$ a *periodic cell* if it is also a cell of the periodic triangulation $\mathrm{Del}(\Gamma X)$ (see Figure 3a).

▶ **Proposition 1** ([12, Lemma 3.4]). *Given a point set X, each cell of $\mathrm{Del}(X_3)$ that has at least one vertex in $\mathrm{dom}^{\LEFTcircle}(0, \Lambda)$ is a periodic cell. Furthermore the set of these cells contains at least one periodic copy of each cell of $\mathrm{Del}(\Gamma X)/\Gamma$.*

After some preprocessing that essentially consists in computing the canonical domain, the first phase internally maintains $\mathrm{Del}(X_3)$ using the CGAL packages for Euclidean Delaunay triangulations [14, 23]. For this we need to develop a systematic way of computing X_3 and obtaining the interface for $\mathrm{Del}(\Gamma X)/\Gamma$ from the internal data structure $\mathrm{Del}(X_3)$.



(a) Cells in blue are guaranteed to be periodic cells by Proposition 1. Green: non-periodic cells. Black vertices: Λ.

(b) The set of canonical cells in blue. Each copy of $\mathrm{dom}(0, \Lambda)$ is labeled with its offset.

(c) Finding the periodic neighbor of σ across the edge τ, which is σ_n'', the canonical representative of σ_n'.

Figure 3 In blue, the points X_3 with their Delaunay triangulation $\mathrm{Del}(X_3)$. The large hexagon is $\mathrm{dom}(0, 3\Lambda)$, while the smaller ones are $\mathrm{dom}(0, \Lambda)$ and (middle, right) its periodic copies. The reduced lattice basis is drawn in red (left, middle).

Once we can guarantee that $\mathrm{Del}(\Gamma X)/\Gamma$ is simplicial and will remain so for any future point insertions, we switch to the second phase. Simpliciality guarantees that the Bowyer-Watson algorithm can be used directly on $\mathrm{Del}(\Gamma X)/\Gamma$. For this purpose we leverage the CGAL machinery for periodic triangulations from the cubic setting [15, 5] and enhance its underlying data structures to work for the generic setting. As we keep only one representative for each vertex, inserting points in this phase is more efficient than in the first phase.

The remainder of this section describes the two phases and the transition between them, as well as the preprocessing of the lattice.

3.1 Lattice preprocessing

We first compute the reduced lattice basis B, which allows us to compute $\mathcal{V}$, the face normals of the canonical domain. We use $\mathcal{V}$ to represent $\mathrm{dom}^{\LEFTcircle}(0, \Lambda)$ and check for containment of a point within $\mathrm{dom}^{\LEFTcircle}(0, \Lambda)$, which helps obtaining the canonical copy of each input point. To find all periodic copies of a canonical point x that lie within $\mathrm{dom}^{\LEFTcircle}(0, 3\Lambda)$, it is sufficient to find all points λ such that the domain $\mathrm{dom}^{\LEFTcircle}(\lambda, \Lambda)$ intersects $\mathrm{dom}^{\LEFTcircle}(0, 3\Lambda)$, and then check $\phi_\lambda(x)$ for containment within $\mathrm{dom}^{\LEFTcircle}(0, 3\Lambda)$ for these points. As the set of these points λ is independent of x, we compute it before phase 1.

Lattice reduction

Many lattice related problems, such as the shortest non-zero vector problem (SVP), are believed to be hard in general [20, 2]. However in low dimensions, we can use two classical iterative algorithms to solve lattice reduction. Let B' be a lattice basis and $B'_{\mathrm{sup}} = \{b'_0, \dots, b'_d\}$ its superbase, as defined in Section 2. Define $c_{ij} := \langle b'_i, b'_j \rangle$.

2-dimensional reduction. A 2-dimensional lattice basis is Lagrange-reduced [13, Section 4.2] if $0 \le 2|c_{12}| \le c_{11} \le c_{22}$. We can negate b'_2 if necessary so that $c_{12} \le 0$. Then $\{b'_1, b'_2, b'_0 := -b'_1 - b'_2\}$ form an obtuse superbase, because $c_{01} = \langle b'_1, -b'_1 - b'_2 \rangle = -c_{11} - c_{12} \le 0$ due to $2|c_{12}| \le c_{11}$, and similarly $c_{02} \le 0$. If a basis is not Lagrange-reduced with $2|c_{12}| > c_{11}$, we exchange b'_2 for a shorter vector that forms a basis with b'_1: Let $b''_2 := b'_2 - sb'_1$, with $s = 1$ if $c_{12} > 0$ and $s = -1$ otherwise. Vector b''_2 is shorter than b'_2 since $c'_{22} = c_{22} + c_{11} - 2sc_{12}$. Since there are only finitely many pairs of lattice vectors within a ball of any given radius, a finite number of applications of this procedure will yield a Lagrange-reduced basis.

3-dimensional reduction. We outline Selling's algorithm [13, Section 4.4], an iterative algorithm that obtains an obtuse superbase in 3 dimensions. In each step of the algorithm, if there is a $c_{ij} > 0$, the algorithm returns a new superbase B''_{sup} defined via $b''_i := -b'_i$, $b''_j := b'_j$, $b''_h := b'_h + b'_i$ and $b''_k := b'_k + b'_i$ where h and k are the remaining two indices different from i and j. For any basis B, let $\sigma(B) := \sum_{b \in B_{\mathrm{sup}}} ||b||^2$. Notice that in each step, $\sigma(B'') = \sigma(B') - 2c_{ij}$. In particular, $\sigma(B'') < \sigma(B')$. This fact, together with the fact that there are only finitely many lattice points in a ball of radius $\sqrt{\sigma(B')}$ (and thus only finitely many quadruplets of vectors whose square magnitudes sum up to at most $\sigma(B')$) guarantees termination of the algorithm.

Intersecting domains

For a given canonical point $x \in X_0$, each of its periodic copies $y \in \Gamma x$ can be obtained as $\phi_\lambda(x)$ for some translation $\phi_\lambda \in \Gamma$. The corresponding lattice point λ can be uniquely written as $\sum_{i=1}^{d} o_i b_i$ for some $o \in \mathbb{Z}^d$ and we write and $\lambda(o) := \lambda$. We then call $o =: o(y)$ the *offset* of y and $\phi_{\lambda(o)}$ the *translation* associated with o. For each point $x \in X_0$ we need to determine the 3^d offsets for which $\phi_{\lambda(o)}(x)$ is within $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$. Fortunately, these offsets have to come from a fixed set of offsets that only depends on the lattice basis. Notice that if $\phi_{\lambda(o)}(\mathrm{dom}^{\triangleleft}(0, \Lambda))$ does not intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$, then $\phi_{\lambda(o)}(x)$ cannot be in $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ for $x \in X_0$. Thus we only need to check those offsets for which $\phi_{\lambda(o)}(\mathrm{dom}^{\triangleleft}(0, \Lambda)) \cap \mathrm{dom}^{\triangleleft}(0, 3\Lambda) \neq \emptyset$.

▶ **Lemma 2.** *There are at most* $4^d - 2^d + 1$ *translates of* $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ *that have non-empty intersection with* $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$.

Proof. If $\mathrm{dom}^{\triangleleft}(\lambda, \Lambda)$ intersects $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ for some $\lambda \in \Lambda$, then λ must be inside $\mathrm{dom}^{\triangleleft}(0, 4\Lambda)$. There are 4^d lattice points within $\mathrm{dom}^{\triangleleft}(0, 4\Lambda)$. For each $v \in \mathcal{V}^+$, the lattice vectors $-2v$ and $2v$ are on the boundary of $\mathrm{dom}^{\triangleleft}(0, 4\Lambda)$, however only $-2v$ is one of those 4^d points within $\mathrm{dom}^{\triangleleft}(0, 4\Lambda)$. For these lattice points $-2v$ on the boundary however the intersection between $\mathrm{dom}^{\triangleleft}(-2v, \Lambda)$ and $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ is empty. Thus only $4^d - |\mathcal{V}^+| = 4^d - (2^d - 1)$ translates of $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$. ◀

In general, we can find this set of offsets via a breadth-first search: We define a graph on Λ with each lattice point λ connected to $\lambda + v$ for $v \in \mathcal{V}$. We start the search at lattice point 0 and terminate once we have found $4^d - 2^d + 1$ offsets, or in the case of non-generic lattices once all 4^d lattice points inside $\mathrm{dom}^{\triangleleft}(0, 4\Lambda)$ have been reached (with the ones on the boundary being discarded).

In 2 dimensions, there is in fact a fixed set S of 13 offsets such that for any lattice, $\mathrm{dom}^{\triangleleft}(\lambda(o), \Lambda)$ only intersects $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ when $o \in S$ (Figure 3b).

▶ **Lemma 3.** *Given a 2-dimensional lattice with basis B and an offset o, if $o \notin \{$(0, 0), (-1, -1), (0, 1), (1, 0), (-1, 0), (0, -1), (1, 1), (-1, -2), (1, 2), (-2, -1), (2, 1), (-1, 1), (1, -1)$\}$, then $\mathrm{dom}^{\triangleleft}(\lambda(o), \Lambda) \cap \mathrm{dom}^{\triangleleft}(0, 3\Lambda) = \emptyset$.*

Proof. Each $\mathrm{dom}^{\triangleleft}(\lambda, \Lambda)$ adjacent to $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ has all but one of its facets in the interior of $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$. Except for those adjacent via one of those facets, all $\mathrm{dom}^{\triangleleft}(\lambda, \Lambda)$ at graph distance 2 from 0 intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$. The ones that don't are exactly the 6 domains $\mathrm{dom}^{\triangleleft}(2v, \Lambda)$ for $v \in \mathcal{V}$. As there are 19 domains at distance at most 2, we have found 13 domains that intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$. As this is the maximum possible number by Lemma 2, we have found all of them. ◀

In 3 dimensions, the same argument yields a set of 51 fixed offsets. However, by Lemma 2, the total number of domains that intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ is 57. Fortunately, the remaining 6 offsets come from a fixed set of 24 offsets that is independent of the lattice.

▶ **Lemma 4.** *There is a set $O_{\leq 2}$ of 51 offsets and a set O_3 of 24 offsets such that for any 3-dimensional lattice with basis B, there is a subset S of O_3 of size 6 such that if $o \notin O_{\leq 2} \cup S$, then $\mathrm{dom}^{\triangleleft}(\lambda(o), \Lambda) \cap \mathrm{dom}^{\triangleleft}(0, 3\Lambda) = \emptyset$.*

Proof. We use the observation that $|\langle v, w\rangle| \leq \langle v, v\rangle$ for any two Voronoi relevant vectors $w, v \in \mathcal{V}$ [13, Section 3.5]. Then the following claim holds:

▷ Claim. Domains $\mathrm{dom}^{\triangleleft}(\nu, \Lambda)$ of graph-distance at most $(k-2)$ from some $\lambda = kv$ for an integer $k \geq 2$ and $v \in \mathcal{V}$ cannot intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$.

Proof. We have $\nu = kv + v_1 + \cdots + v_{k-2}$ for some $v_i \in \mathcal{V}, i = 1, \ldots, k-2$. Then $\langle w, v\rangle = k\langle v, v\rangle + \langle v, v_1\rangle + \cdots + \langle v, v_{k-2}\rangle \geq k\langle v, v\rangle - (k-2)\langle v, v\rangle = 2\langle v, v\rangle$ due to the observation above. This means that ν is not in the interior of $\mathrm{str}(4v)$ and thus the interior of the 4-scaled domain. Therefore $\mathrm{dom}^{\triangleleft}(\nu, \Lambda) \cap \mathrm{dom}^{\triangleleft}(0, 3\Lambda) = \emptyset$. ◁

For a generic 3-dimensional lattice, the Voronoi relevant vectors correspond to the non-empty subsets of B_{sup} of size at most 3, independent of the actual vectors of B_{sup}. Thus the graph we defined on Λ is isomorphic for all generic lattices. For non-generic lattices it is isomorphic to a subgraph of the graph for generic lattices; thus it is sufficient to restrict our attention to generic lattices.

We enumerate all lattice points at graph distance 3 from the origin. For a generic lattice, this yields 110 points, however all but 24 of them can be written as $3v$ or $3v + w$ for Voronoi relevant vectors v and w. The set of the 24 corresponding offsets is O_3, while $O_{\leq 2}$ are those 51 offsets that the argument from Lemma 3 yields. All lattice points at graph distance 4 can be written as $4v + kw$ for $k \leq 2$ and Voronoi relevant vectors v and w, and thus their domains and consequently any domains at higher distances cannot intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$. ◀

3.2 Phase 1

Phase 1 maintains the Euclidean Delaunay triangulation of X_3. For each new point to be inserted, we first find its canonical copy. Then we can compute its periodic copies that are contained within $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ using Lemmas 3 and 4. These are then inserted into $\mathrm{Del}(X_3)$. To provide user access to the cells of $\mathrm{Del}(\Gamma X)/\Gamma$, we define a notion of canonical cell in $\mathrm{Del}(X_3)$ to get a representative for each cell of $\mathrm{Del}(\Gamma X)/\Gamma$.

Canonical points

For each point of X, finding its periodic copy that lies in $\mathrm{dom}^{\lhd}(0,\Lambda)$ is equivalent to solving the closest vector problem (CVP), i.e. given $x \in X$, determining the lattice point that is closest to x. For arbitrary dimensions this problem is known to be NP-hard [20]. For the exact version of CVP, various iterative algorithms have been described [1, 19, 16]. As we are only operating in 2 and 3 dimensions, any of them would suffice for us in practice in terms of running time, and we will describe the algorithm by Sommer et al [19] due to its simplicity.

For a real number r, define $\mathrm{round}(r)$ to be the closest integer to r. If this integer is not unique, then it is the one with the smallest absolute value. This definition ensures convergence in cases where x is on the boundary of a Dirichlet domain of the lattice [19]. Recall that $\mathcal{V}^+$ is a set of normals of the facets of the canonical domain, and can be obtained from a reduced basis B (see Section 2). We first sort the vectors of $\mathcal{V}^+$ by their magnitude. As $\mathrm{dom}^{\lhd}(0,\Lambda)$ is the intersection of the strips $\mathrm{str}(v)$ for $v \in \mathcal{V}^+$, we need to find the periodic copy of x that is in all these $\mathrm{str}(v)$. We loop through the vectors v of $\mathcal{V}^+$ and for each of them perform the following operations: Compute $c := \frac{\langle x,v\rangle}{\langle v,v\rangle}$. If $-0.5 \le c < 0.5$, then already $x \in \mathrm{str}(v)$. If not, then we subtract $\mathrm{round}(c) \cdot v$ from x. Note that after such a step, it is guaranteed that $x \in \mathrm{str}(v)$. However after modifying x for a longer vector v, it might happen that x is moved outside of $\mathrm{str}(v')$ for some shorter vector v' again. Therefore we need to repeatedly loop through $\mathcal{V}^+$ and perform this operation until $x \in \mathrm{str}(v)$ for all $v \in \mathcal{V}^+$.

Notice that in each step where x is modified, the magnitude of x strictly decreases. Therefore this algorithm terminates in finite time, as the number of lattice vectors within a ball of given radius $||x||$ is finite.

Extracting representative cells

Recall that our interface specifies access to the cells (and lower-dimensional simplices) of $\mathrm{Del}(\Gamma X)/\Gamma$. Thus for each of its cells we have to provide one representative from $\mathrm{Del}(X_3)$. For a cell σ of either $\mathrm{Del}(\Gamma X)$ or $\mathrm{Del}(X_3)$, with $V(\sigma) = \{x_1, \ldots, x_d\}$, we define its offset as the vector $o(\sigma) = \min_{x \in V(\sigma)}\{o(x)\}$ where the minimum is taken lexicographically. Note that this definition differs from [4, Convention 3.3.1] where the coordinate-wise minimum is taken. If the offset of a cell is the 0-vector, we call it a *canonical cell*. Note that due to our definition a canonical cell always has a vertex with offset 0. Therefore by Proposition 1 a canonical cell of $\mathrm{Del}(X_3)$ is also periodic , see Figure 3b. For $\sigma \in \mathrm{Del}(\Gamma X)$, the translated cell $\sigma - \sum_{i=1}^{d} o_i b_i$ is called its *canonical representative*. This means that for each class of cells in $\mathrm{Del}(\Gamma X)$ (or equivalently each cell of $\mathrm{Del}(\Gamma X)/\Gamma$), there is a unique canonical representative in $\mathrm{Del}(X_3)$. Therefore we can get a set of representative cells by iterating over the cells of $\mathrm{Del}(X_3)$ and selecting those that are canonical.

Neighborhood relations

In accordance with our interface, we need to provide neighborhood relations for the vertices and cells of $\mathrm{Del}(\Gamma X)/\Gamma$. These vertices and cells are represented by the canonical vertices and cells of $\mathrm{Del}(X_3)$, whose neighbors in $\mathrm{Del}(X_3)$ (which we have access to) may differ from the neighbors in $\mathrm{Del}(\Gamma X)/\Gamma$ (which we want to find). We outline how to get the neighors of a cell of $\mathrm{Del}(\Gamma X)/\Gamma$ from $\mathrm{Del}(X_3)$, and note that other neighborhood relations work in a conceptually similar way. Note that we do not store these relations explicity, but we compute them upon request and may cache them for future access.

Consider a canonical cell σ of $\mathrm{Del}(X_3)$ and a neighboring cell σ_n. If σ_n is canonical, it is the neighbor of σ in $\mathrm{Del}(\Gamma X)$. If σ_n is not canonical but has a vertex in $\mathrm{dom}^{\lhd}(0,\Lambda)$, then it is periodic and we return the canonical representative of this cell. However, it is possible that all vertices of σ_n are outside $\mathrm{dom}^{\lhd}(0, \Lambda)$, and thus σ_n might not be a periodic cell at all. In that case, we need to consider the facet τ separating σ and its neighbor σ_n. As τ is a facet of a canonical cell, it is also a periodic facet (i.e. a facet of $\mathrm{Del}(\Gamma X)$). We compute τ's offset $o := o(\tau)$. Then $\tau' := \phi_{\lambda(-o)}(\tau)$ is the canonical representative of τ and $\sigma' := \phi_{\lambda(-o)}(\sigma)$ as well as its neighbor σ'_n across τ' are periodic cells because they share τ', which has a vertex in $\mathrm{dom}^{\lhd}(0, \Lambda)$. If σ'_n is canonical, then is it the canonical representative of the neighbor of σ in $\mathrm{Del}(\Gamma X)$; if not, then its canonical representative is. Figure 3c illustrates this process.

In practice, we store the canonical representative of each vertex of X_3. To obtain the canonical representative of a cell (or facet) σ, we need to choose a vertex x whose offset is minimal (lexicographically) among its vertices. This ensures that the periodic copy x' of x in the canonical version of the cell is inside $\mathrm{dom}^{\lhd}(0, \Lambda)$. Then one of the cells (or facets) incident to x' in $\mathrm{Del}(X_3)$ is the canonical representative of σ.

3.3 Transition

Phase 2 is more efficient than phase 1 as it directly operates on $\mathrm{Del}(\Gamma X)/\Gamma$; however, we cannot use it from the start as the Bowyer-Watson algorithm [3, 21] comes with some constraints. The Bowyer-Watson algorithm is an incremental algorithm inserting points one by one. For each new point x, it determines the *conflict zone*, which is the set of cells whose circumsphere contains x. All these cells are removed, and all boundary facets of the resulting hole are connected to x to fill in the hole with new cells. The algorithm requires this hole to be a topological d-ball, which is not always guaranteed for $\mathrm{Del}(\Gamma X)/\Gamma$. However the following criterion is a sufficient condition for the Bowyer-Watson algorithm to work [7].

▶ **Lemma 5** ([7, Criterion 3.11])**.** *If for every cell in* $\mathrm{Del}(\Gamma X)/\Gamma$ *the circumradius is smaller than* $\frac{1}{4}\mathrm{sv}(\Lambda)$*, then* $\mathrm{Del}(\Gamma X')/\Gamma$ *is simplicial for every* $X' \supseteq X$*.*

If this criterion is fulfilled, we can safely switch to phase 2. To detect at which point in phase 1 the criterion is fulfilled, we maintain a set S_{big} of *big* cells, which are canonical cells whose circumradius is larger than or equal to $\frac{1}{4}\mathrm{sv}(\Lambda)$. We update S_{big} during each point insertion.

Assume we wish to insert a new point x into the periodic triangulation of some point set X. On a high level, we need to remove the big cells in the conflict zone of x from S_{big}, and then add the newly created big cells to S_{big}. In practice, we have already computed the triangulation $\mathrm{Del}(X_3)$ and have to insert the periodic copies of x that are within $\mathrm{dom}^{\lhd}(0, 3\Lambda)$, i.e. $\Gamma x \cap \mathrm{dom}^{\lhd}(0, 3\Lambda)$, into $\mathrm{Del}(X_3)$. We first detect the conflict zone of x_0, the canonical copy of x. For each big cell in the conflict zone we check if it has a canonical copy in S_{big}, and remove that copy from S_{big} if it does. While the conflict zone may contain non-periodic cells, Lemma 6 guarantees that we capture a representative of each cell from $\mathrm{Del}(\Gamma X)$ that is in conflict with x. Next we insert all the copies $\Gamma x \cap \mathrm{dom}^{\lhd}(0, 3\Lambda)$ into $\mathrm{Del}(X_3)$. Finally, for each big cell incident to x_0 in the resulting triangulation, we add its canonical copy to S_{big}. Note that all these cells have a canonical copy by Proposition 1 as they have a vertex inside $\mathrm{dom}^{\lhd}(0, \Lambda)$.

▶ **Lemma 6.** *Let C be the conflict zone of some point $x \in \mathrm{dom}^{\lhd}(0, \Lambda)$ with respect to* $\mathrm{Del}(\Gamma X)$*. Then the conflict zone of x with respect to* $\mathrm{Del}(X_3)$ *contains at least one periodic copy of each cell from C.*

Proof. First observe that if a cell of $\mathrm{Del}(\Gamma X)$ has its circumcenter at the origin, then all its vertices must be within $\mathrm{dom}(0, \Lambda)$, because if the circumsphere contains a point outside $\mathrm{dom}(0, \Lambda)$, then it also contains its canonical copy. Via translation it follows that if a cell has its circumcenter in $\mathrm{dom}^{\triangleleft}(0, 2\Lambda)$, then its vertices are in $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ and thus it is a cell of $\mathrm{Del}(X_3)$. So assume we have a cell σ in C whose circumcenter c is not within $\mathrm{dom}^{\triangleleft}(0, 2\Lambda)$. Then there is a facet of $\mathrm{dom}^{\triangleleft}(0, 2\Lambda)$ with respect to which c is outside. Let f be its face normal. Then $x + f$ is also contained in the circumsphere of σ. Reversely, $\sigma - f$ is a cell whose circumsphere contains x, and furthermore its circumcenter is closer to 0 than c, see Figure 4. As there are only finitely many periodic copies of c within a given distance from 0, after

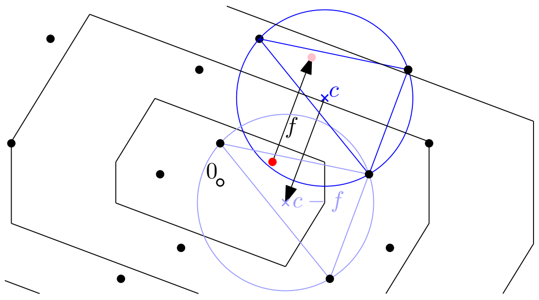


Figure 4 The red point x is in the conflict zone of the blue cell σ, which is not a cell of $\mathrm{Del}(X_3)$ and whose circumcenter c is outside $\mathrm{dom}^{\triangleleft}(0, 2\Lambda)$. However the light blue cell $\sigma - f$ is a periodic copy of σ that is in $\mathrm{Del}(X_3)$ and x is in its conflict zone.

applying this process a finite number of times we eventually obtain a cell whose circumsphere contains x and whose circumcenter is in $\mathrm{dom}^{\triangleleft}(0, 2\Lambda)$. This cell is a periodic copy of σ, is contained in $\mathrm{Del}(X_3)$ and thus also part of the conflict zone of x with respect to $\mathrm{Del}(X_3)$. ◀

Once S_{big} is empty, the criterion of Lemma 5 is fulfilled, and we can internally convert our triangulation from $\mathrm{Del}(X_3)$ to $\mathrm{Del}(\Gamma X)/\Gamma$, which is maintained in phase 2. We initialize the periodic triangulation data structure with the set of canonical vertices X_0 and canonical cells obtained from $\mathrm{Del}(X_3)$, as well as the adjacency and incidence relations outlined earlier.

3.4 Phase 2

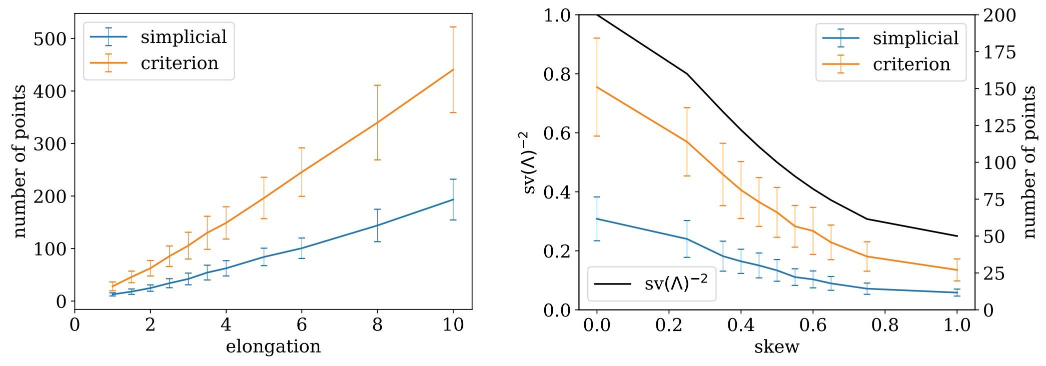
Phase 2 operates directly on the torus triangulation $\mathrm{Del}(\Gamma X)/\Gamma$. Thus it maintains only one copy of each cell and vertex and point insertion is faster than in phase 1. The data structure it uses to represent $\mathrm{Del}(\Gamma X)/\Gamma$ is akin to the one used in CGAL for the cubic case, and closely resembles the interface we defined for our algorithm: Only X_0 is stored as vertex set, and each cell is represented by its vertices, with a vertex encoded as a pair (x, o) of a point $x \in X_0$ and an offset so that its geometric location is $\phi_{\lambda(o)}(x)$. While in the cubic case each offset coordinate either takes the value 0 or 1 and offsets can be encoded in d bits, in our more general setting offsets can take any of the lattice-specific values from Lemma 2. Unlike in phase 1, most neighborhood relations for $\mathrm{Del}(\Gamma X)/\Gamma$ are already stored explicitly in the data structure: For each cell its adjacent cells are stored, and for each vertex one incident cell is stored. The remaining neighborhood relations required by our interface are obtained implicitly from the explicitly stored ones. With each point insertion, all stored neighborhood relations have to be updated accordingly.

Point insertion. To insert a new point x into $\mathrm{Del}(\Gamma X)/\Gamma$ using the Bowyer-Watson algorithm, first we compute the canonical copy x_0. Then we locate the cell containing x_0, via a traversal starting from an arbitrary cell. The conflict zone is computed via a search starting in the cell containing the point x_0. Whenever we are traversing cells, care has to be taken to maintain the correct offset of the affected cells relative to x_0, and similarly when creating new cells to fill in the hole left by the deleted conflict zone. For the cubic case the details are described in [4, Section 3.3], and we omit the technical adjustments needed to make these steps work in the more general case.

4 Experimental results

Points until transition. We experimentally evaluated the number of points required until the criterion of Lemma 5 is fulfilled and the transition to phase 2 occurs. Lemma 5 requires all of $\mathbb{R}^d/\Gamma$ to be covered by the balls of radius $\frac{1}{4}\mathrm{sv}(\Lambda)$ around X_0. As the volume of the torus equals the volume of the Dirichlet domain of 0, denoted as $\mathrm{vol}(\mathrm{dom}(0,\Lambda))$, we expect the number of points until switching to phase 2 to be roughly proportional to $\mathrm{vol}(\mathrm{dom}(0,\Lambda))/\mathrm{sv}(\Lambda)^d$, assuming the points are sampled uniformly at random.

To investigate this in 2 dimensions, we parametrize a 2-dimensional space of lattices. We call the parameters the elongation ℓ and the skew s. The basis of the lattice with elongation ℓ and skew s is $b_1 = (\ell, 0)$ and $b_2 = (s \cdot \ell/2, 1)$. With $\ell \geq 1$ and s between 0 and 1 we can parametrize all 2-dimensional lattices up to symmetry and scaling. Note that the skew affects $\mathrm{sv}(\Lambda)$ but not $\mathrm{vol}(\mathrm{dom}(0,\Lambda))$, while the elongation is proportional to $\mathrm{vol}(\mathrm{dom}(0,\Lambda))$ but does not affect the $\mathrm{sv}(\Lambda)$. Fixing the skew at 0 and varying the elongation (Figure 5a), we see that the number of random points needed until the phase switch appears to be proportional to the elongation. The same applies to the number of points until the resulting triangulation is simplicial for the first time. Figure 5b shows the same statistics for lattices of fixed area but varying skew. In addition we plot the inverse of $\mathrm{sv}(\Lambda)^2$ for comparison, and observe that it behaves similarly albeit not entirely proportionally.



(a) Results for fixed skew $s = 0$ and varying elongation ℓ.

(b) Results for $\ell = 4$ and varying s. The inverse of the square length of the shortest non-zero lattice vector in black, with its y-axis on the right.

Figure 5 For different lattices, the number of points inserted into a periodic triangulation until (blue) $\mathrm{Del}(\Gamma X)/\Gamma$ is simplicial for the first time, and (orange) $\mathrm{Del}(\Gamma X)/\Gamma$ fulfills Lemma 5. Each data point is the mean of 200 trials, and the bars represent the standard deviation.

In 3-dimensions, a parametrization of lattices up to symmetry and scaling needs 5 parameters, and thus an analysis like in 2D is impractical. As mentioned before, we expect that the number of points until the switch to phase 2 is proportional to $\mathrm{vol}(\mathrm{dom}(0,\Lambda))/\mathrm{sv}(\Lambda)^3$. By comparing the two, Figure 6 confirms this relationship for 182 lattices whose basis vectors have randomly sampled direction and a random magnitude between 1 and 100.

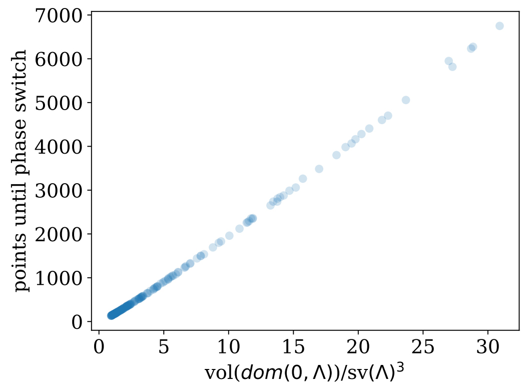


Figure 6 Phase switching in 3D: Each point in the plot represents a lattice, with $\mathrm{vol}(\mathrm{dom}(0,\Lambda))/\mathrm{sv}(\Lambda)^3$ on the x-axis and the number of points inserted until the switch to phase 2 occurred on the y-axis. Each y-value is obtained as the average over 200 point sets, each distributed uniformly at random.

Running times. We evaluate the running times of our algorithm for different 3-dimensional lattices, and compare them to the existing CGAL implementations. The data points collected are from Delaunay triangulations of 10^k uniformly sampled random points for k up to 7. Each data point is an average of 300 trials (10 trials for 10^7 points). The experiments were conducted on a laptop running Fedora 30 64-bits, with two 6-core Intel(R) i9-8950HK CPU clocked at 2.90GHz, and with 32GB of RAM. The CGAL kernel used was CGAL::Exact_predicates_inexact_constructions_kernel and CPU time was measured using CGAL's timer tools. The code was compiled using clang 8.0.0 with compilation flags -O3 and -DNDEBUG.

Table 1 shows a comparison between the CGAL implementation of Euclidean Delaunay triangulations [14] (with random points uniformly sampled in the unit cube), periodic Delaunay triangulations in the cubic setting [5] and our algorithm for various lattices, including the cubic lattice. For each lattice, we also measured the average number of points until the switch to phase 2. As our algorithm and the one for the cubic setting are based on the same code base when operating directly on the torus triangulation, their runtimes are comparable for large point sets. It should be noted that when similar experiments were conducted in 2010 to compare the Euclidean and cubic periodic algorithms [4, Section 3.6.2], both were performing comparably. Since then, Euclidean Delaunay triangulations in CGAL have seen significant optimizations, which were not applied to periodic triangulations. This also explains why our algorithm is faster than the cubic periodic algorithm for point sets where phase 1 takes up a significant portion of the running time, as internally we use a Euclidean rather than a periodic triangulation.

5 Discussion

Weighted points. Phase 1 of our algorithm readily generalizes to weighted point sets. In particular, Proposition 1 still holds for weighted point sets (see Appendix A for a proof). Phase 2 only works for weighted points under additional restrictions because $\mathrm{Del}(\Gamma X)/\Gamma$ can not be guaranteed to remain simplicial, in particular after inserting points with large weights. Such point insertions that break simpliciality can be prevented by requiring points to be inserted in decreasing order of weight, or like in the cubic implementation in CGAL by severely restricting the range of weights a point can have. Thus an implementation is subject to a tradeoff between flexibility (phase 1) and performance (phase 2).

Table 1 Running time (in seconds) of various Delaunay triangulation algorithms on random point sets of different sizes. Our algorithm ("Lattice") is evaluated for different lattices including the cubic lattice, face-centered cubic (FCC) lattice and two other lattices Λ_1 and Λ_2 with bases $B_1 = \{(0.5, -0.5, 0.1), (-0.5, 0.5, 0.1), (0.5, 0.5, -0.1)\}$ and $B_2 = \{(1, 0, 0), (-0.5, \sqrt{3}/2, 0), (0, 0, 0.05)\}$. For each lattice we also record the average number of points until the switch to phase 2 (n_{switch}).

Algorithm	Euclidean [14]	Cubic [5]	Lattice			
Lattice	–	Cubic	Cubic	FCC	Λ_1	Λ_2
$\frac{\text{vol}(\text{dom}(0,\Lambda))}{\text{sv}(\Lambda)^3}$	–	1.00	1.00	0.71	12.50	346.41
n_{switch}	–	145 [4]	141	94	2519	89950
10^0	0.0000	0.0001	0.0002	0.0001	0.0003	0.0004
10^1	0.0000	0.0160	0.0033	0.0026	0.0044	0.0035
10^2	0.0004	0.1848	0.0461	0.0287	0.0460	0.0380
10^3	0.0049	0.5957	0.0858	0.0446	0.9812	0.6372
10^4	0.0487	0.9591	0.3832	0.1642	4.6602	16.5759
10^5	0.5679	4.8119	4.5153	2.7868	10.8139	362.7956
10^6	6.5974	93.9327	95.1447	51.5945	58.1568	517.9715
10^7	59.5152	2314.3618	2317.7867	1215.2648	1799.4515	2983.0943

Higher dimensions. Conceptually, our algorithm generalizes to higher dimensions. The only step that is dimension-dependent and therefore requires significant adjustments is obtaining a representation of $\text{dom}^{\mathbb{C}}(0, \Lambda)$. While up to 3 dimensions every lattice has an obtuse superbase, this does not hold in higher dimensions. Therefore we would need to find a different way of obtaining the set of Voronoi relevant vectors. Complexity-wise, computing the canonical representative of a point is believed to be hard. Furthermore, the number of copies that we compute of each input point in phase 1 is 3^d. Therefore we expect the runtime of our algorithm to be exponential in the dimension, but still feasible in practice as long as the dimension is not too large.

Dummy points. For best performance, transition from phase 1 to phase 2 should occur as soon as possible. Intuitively, this happens when the input point distribution does not have large gaps. We can achieve this by first inserting additional *dummy points*, which are removed at the end (if possible) [7]. In practice, we can choose these points from a sufficiently fine hexagonal lattice (or body-centered cubic in 3 dimensions), such that the open spheres of radius $\frac{1}{4}\text{sv}(\Lambda)$ around the points cover the entire torus.

Software distribution. We aim to provide a CGAL package for periodic Delaunay triangulations for arbitrary lattices in 2 and 3 dimensions. The current state of our implementation is available online (see Footnote 1). Both the 2- and 3-dimensional implementations have been integrated into the CGAL codebase and only require some additional refactoring and optimizing to adhere to CGAL's quality standards. Automated tests and documentation still have to be produced. An extension to weighted Delaunay triangulations, referred to as regular triangulations in CGAL, is planned for the future.

References

1 Erik Agrell, Thomas Eriksson, Alexander Vardy, and Kenneth Zeger. Closest point search in lattices. *IEEE Trans. Inform. Theory*, 48(8):2201–2214, 2002. doi:10.1109/TIT.2002.800499.

2 Miklós Ajtai. The shortest vector problem in L2 is NP-hard for randomized reductions. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 10–19, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276705.

3 Adrian Bowyer. Computing Dirichlet tessellations. *The computer journal*, 24(2):162–166, 1981.

4 Manuel Caroli. *Triangulating Point Sets in Orbit Spaces*. PhD thesis, Université Nice Sophia Antipolis, 2010. URL: `https://tel.archives-ouvertes.fr/tel-00552215`.

5 Manuel Caroli, Aymeric Pellé, Mael Rouxel-Labbé, and Monique Teillaud. 3D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0.2 edition, 2020. URL: `http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic3Triangulation3`.

6 Manuel Caroli and Monique Teillaud. 3D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.5 edition, 2009.

7 Manuel Caroli and Monique Teillaud. Delaunay triangulations of closed Euclidean d-orbifolds. *Discrete & Computational Geometry*, 55(4):827–853, 2016. URL: `https://hal.inria.fr/hal-01294409`, `doi:10.1007/s00454-016-9782-6`.

8 J. H. Conway and N. J. A. Sloane. Low-dimensional lattices VI: Voronoi reduction of three-dimensional lattices. *Proc. R. Soc. Lond. A*, pages 55–68, 1992.

9 B. Delaunay. Sur la sphère vide. À la mémoire de Georges Voronoï. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 7:793–800, 1934. URL: `http://mi.mathnet.ru/eng/izv4937`.

10 Vincent Despré, Jean-Marc Schlenker, and Monique Teillaud. Flipping geometric triangulations on hyperbolic surfaces. In *Proceedings of the Thirty-sixth International Symposium on Computational Geometry*, 2020. To appear. Preliminary version: `https://hal.inria.fr/hal-02400219`.

11 Olivier Devillers and Monique Teillaud. Perturbations for Delaunay and weighted Delaunay 3D triangulations. *Computational Geometry: Theory and Applications*, 44:160–168, 2011. URL: `http://hal.inria.fr/inria-00560388/`, `doi:10.1016/j.comgeo.2010.09.010`.

12 Nikolai Dolbilin and Daniel Huson. Periodic Delone tilings. *Periodica Mathematica Hungarica*, 34:1-2:57–64, 1997.

13 Peter Engel. *Geometric crystallography: an axiomatic introduction to crystallography*. Springer, Dordrecht, 1986. `doi:10.1007/978-94-009-4760-3`.

14 Clément Jamin, Sylvain Pion, and Monique Teillaud. 3D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0.2 edition, 2020. URL: `https://doc.cgal.org/latest/Manual/packages.html#PkgTriangulation3`.

15 Nico Kruithof. 2D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.4 edition, 2014. URL: `http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic2Triangulation2Summary`.

16 Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM J. Comput.*, 42(3):1364–1391, 2013. `doi:10.1137/100811970`.

17 The CGAL Project. URL: `http://www.cgal.org`.

18 Chris Rycroft. Voro++: A three-dimensional Voronoi cell library in c++. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2009. URL: `http://math.lbl.gov/voro++/`.

19 Naftali Sommer, Meir Feder, and Ofir Shalvi. Finding the closest lattice point by iterative slicing. *SIAM J. Discrete Math.*, 23(2):715–731, 2009. `doi:10.1137/060676362`.

20 Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, Mathematische Instituut, Uni. Amsterdam Report, April 1981.

21 David F Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The computer journal*, 24(2):167–172, 1981.

22 Thomas F Willems, Chris H Rycroft, Michaeel Kazi, Juan C Meza, and Maciej Haranczyk. Algorithms and tools for high-throughput geometry-based analysis of crystalline porous materials. *Microporous and Mesoporous Materials*, 149(1):134–141, 2012. URL: `http://zeoplusplus.org/`.

23 Mariette Yvinec. 2D triangulation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0.2 edition, 2020. URL: `https://doc.cgal.org/latest/Manual/packages.html#PkgTriangulation2`.

A Generalization to weighted points

For a weighted point x, we denote its weight with w_x. For weighted points $x, y \in X$, their power distance is $\mathrm{dist}(x, y) := \|x - y\| - w_x - w_y$. For an arbitrary point $p \in \mathbb{R}$ that is not part of X we define the power distance from x to p as $\mathrm{dist}(x, p) := \|x - p\| - w_x$ (as if the weight of p was 0).

We call two points x and y orthogonal if $\mathrm{dist}(x, y) = 0$. For $d + 1$ points $Q \subseteq X$ (if in general position) there is a unique point z with weight w_z that is orthogonal to all points of Q. This point is called the orthocenter or power sphere of Q. If Q is a set of points forming a cell (simplex) in the Delaunay triangulation, then for all points $x \in Q$ and $y \in X \setminus Q$ it holds that $\mathrm{dist}(y, z) > \mathrm{dist}(x, z) = 0$. We call this the empty-sphere property. The converse holds as well. The perpendicular bisector of two weighted points x and y is the set of points p with $\mathrm{dist}(x, p) = \mathrm{dist}(y, p)$.

▶ **Proposition 7** (Generalization of Lemma 3.2–3.4 from [12]). *Given a set X of representatives for our point set, let $X_3 := \mathrm{dom}^{\triangleleft}(0, 3\Lambda) \cap \Gamma X$, i.e. all periodic copies of these points that lie within the Dirichlet domain of 0 scaled by a factor of 3. Let T_0 be those cells of $\mathrm{Del}(X_3)$ that have at least one vertex in $\mathrm{dom}^{\triangleleft}(0, \Lambda)$. Then the cells of T_0 are all part of the triangulation of ΓX. Furthermore, these cells contain at least one representative of each class of cells from $\mathrm{Del}(\Gamma X)$.*

Proof. We will prove the proposition in 3 steps.

▷ Claim 1. Assume one of our points is $x_0 = 0$ (with arbitrary weight). Then the orthocenter c_T of any Delaunay cell that has x_0 as a vertex is within $\mathrm{dom}(0, \Lambda)$.

Proof. Assume not. Then there is a face F_i of $\mathrm{dom}(0, \Lambda)$ such that F_i separates x_0 from c_T. Let f_i be the corresponding translation lattice vector orthogonal to F_i. Then $x_0 + f_i$ is strictly closer to c_T than x_0 because it has the same weight as x_0. This is a contradiction to the empty-sphere property. ◁

▷ Claim 2. For a cell containing x_0 as a vertex, all vertices are in $\mathrm{dom}(0, 2\Lambda)$.

Proof. Assume some vertex x is not. Consider the perpendicular bisector between x and $x - f_i$ where f_i is the face normal vector orthogonal to the face F_i of $\mathrm{dom}(0, \Lambda)$ with respect to which x is outside of $\mathrm{dom}(0, 2\Lambda)$. This bisector is separated from x_0 and the orthocenter c_T by F_i, the face of $\mathrm{dom}(0, \Lambda)$ that is parallel to this bisector. In particular, it follows from Claim 1 that c_T is (strictly) on the $x - f_i$ side of the perpendicular bisector. So $x - f_i$ is closer to c_T than x and thus the empty-sphere property is violated. See Figure 7 for reference. ◁

▷ Claim 3. All cells σ having a vertex in $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ are entirely contained in $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$.

Proof. Let x be a vertex of σ that is within $\mathrm{dom}^{\triangleleft}(0, \Lambda)$. Shift the entire point set ΓX and its triangulation by the vector $-x$ so that x now coincides with 0. Now from Claim 2 it follows that the other vertices of the shifted cell are within $\mathrm{dom}(0, 2\Lambda)$. Adding the vector $+x$ to these shifted vertices we get back to the original setting, but because $x \in \mathrm{dom}^{\triangleleft}(0, \Lambda)$ we also know now that these vertices are within $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$, as $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$ is the Minkowski sum of $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ and $\mathrm{dom}(0, 2\Lambda)$. ◁

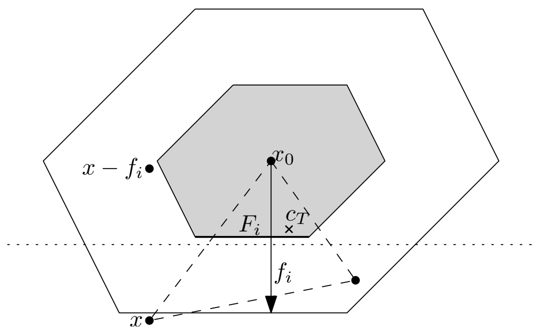


Figure 7 A cell (dashed) with a vertex x that is outside of $\mathrm{dom}(0, 2\Lambda)$. The dotted line is the perpendicular bisector between x and $x - f_i$.

Every cell of $\mathrm{Del}(\Gamma X)$ that has a vertex in $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ has all its vertices in X_3. Furthermore, because it fulfils the empty-sphere property in ΓX, then it also fulfils this property in $X_3 \subset \Gamma X$, and thus is present in $\mathrm{Del}(X_3)$. As every point from X has a representative in $\mathrm{dom}^{\triangleleft}(0, \Lambda)$, also each cell from $\mathrm{Del}(\Gamma X)$ has a representative in T_0, proving the statement. ◀

B Detailed proof of Lemma 4

▶ **Lemma 4** (extended version). *Let* $O_3 :=$ {*(3, 2, 1), (2, 1, -1), (3, 1, 2), (2, -1, 1), (1, -1, -2), (1, -2, -1), (2, 3, 1), (1, 2, -1), (1, 3, 2), (-1, 2, 1), (-1, 1, -2), (-2, 1, -1), (2, 1, 3), (1, -1, 2), (1, 2, 3), (-1, 1, 2), (-1, -2, 1), (-2, -1, 1), (-1, -2, -3), (-1, -3, -2), (-2, -1, -3), (-3, -1, -2), (-2, -3, -1), (-3, -2, -1)*}

and $O_{\leq 2} :=$ {*(0, 0, 0), (-1, -1, -1), (0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 0), (1, 0, 1), (0, -1, -1), (0, 1, 1), (-1, 0, -1), (-1, -1, 0), (1, 1, 1), (0, 0, -1), (0, -1, 0), (-1, 0, 0), (1, 2, 0), (1, 0, 2), (-1, -2, -2), (2, 1, 0), (2, 0, 1), (1, -1, -1), (0, 1, 2), (-2, -1, -2), (0, 2, 1), (-1, 1, -1), (-2, -2, -1), (-1, -1, 1), (1, 1, 2), (-1, -1, -2), (1, 2, 1), (0, 1, -1), (-1, -2, -1), (0, -1, 1), (2, 1, 1), (1, 0, -1), (1, -1, 0), (-2, -1, -1), (-1, 0, 1), (-1, 1, 0), (1, 2, 2), (-1, 0, -2), (-1, -2, 0), (2, 1, 2), (0, -1, -2), (2, 2, 1), (1, 1, -1), (0, -2, -1), (1, -1, 1), (-2, -1, 0), (-2, 0, -1), (-1, 1, 1)*}.

Given a 3-dimensional lattice and an offset o, there is a subset S of size 6 of O_3 such that if $o \notin O_{\leq 2} \cup S$, then $\mathrm{dom}^{\triangleleft}(\lambda(o), \Lambda) \cap \mathrm{dom}^{\triangleleft}(0, 3\Lambda) = \emptyset$.

Proof. As mentioned in the proof of Theorem 3.5 from [13], $\frac{v}{2}$ is on the boundary of $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ for every Voronoi-relevant vector v.

▷ Claim 1. The orthogonal projection of v into the 1-dimensional subspace spanned by another Voronoi relevant vector $w \in \mathcal{V}^+$ has magnitude less than w.

Proof. All facets of $\mathrm{dom}^{\triangleleft}(0, \Lambda)$ are contained in $\mathrm{str}(w)$ for all $w \in \mathcal{V}^+$, so because $\frac{v}{2}$ is on the boundary of $\mathrm{dom}^{\triangleleft}(0, \Lambda)$, it is contained in $\mathrm{str}(w)$. By definition of $\mathrm{str}(w)$ this implies $|\langle \frac{1}{2} v, w\rangle / \langle w, w\rangle| \leq \frac{1}{2}$, from which it follows that $|\langle v, w\rangle| \leq \langle w, w\rangle$. ◁

▷ Claim 2. Domains $\mathrm{dom}^{\triangleleft}(\nu, \Lambda)$ of graph-distance $(k-2)$ from some $\lambda = kv$ for an integer k and $v \in \mathcal{V}$ cannot intersect $\mathrm{dom}^{\triangleleft}(0, 3\Lambda)$.

Proof. We have $\nu = kv + v_1 + \cdots + v_{k-2}$ for some $v_i \in \mathcal{V}$. Then $\langle w, v\rangle = k\langle v, v\rangle + \langle v, v_1\rangle + \cdots + \langle v, v_{k-2}\rangle \geq k\langle v, v\rangle - (k-2)\langle v, v\rangle = 2\langle v, v\rangle$ due to Claim 1. This means that ν is not in the interior of $\mathrm{str}(4v)$ and thus the interior of the 4-scaled domain. Therefore $\mathrm{dom}^{\triangleleft}(\nu, \Lambda) \cap \mathrm{dom}^{\triangleleft}(0, 3\Lambda) = \emptyset$. ◁

Let $B_{\text{sup}} = \{b_0, b_1, b_2, b_3\}$. Note that because $b_0 = -(b_1 + b_2 + b_3)$, every lattice point λ can be written as a non-negative integer combination of three of these extended basis vectors, i.e. $\lambda = c_1 a + c_2 b + c_3 c$ with $c_i \in \mathbb{Z}, c_1 \geq c_2 \geq c_3 \geq 0$ and $a, b, c \in B_{\text{sup}}$. This representation is unique up to permutation of basis vectors with the same coefficient.

Enumerating all $\operatorname{dom}^{◖}(\lambda, \Lambda)$ at graph distance 3 from $\operatorname{dom}^{◖}(0, \Lambda)$, we get lattice points λ with non-negative integer combinations of the following types:

$$\begin{aligned}
&3a \\
&3a + b &&= 3a &&+ b \\
&3a + b + c &&= 3a &&+ (b + c) \\
&3a + 2b &&= 3(a + b) &&- b \\
&3a + 2b + c \\
&3a + 2b + 2c &&= 3(a + b + c) &&- (b + c) \\
&3a + 3b \\
&3a + 3b + c &&= 3(a + b) &&+ c \\
&3a + 3b + 2c &&= 3(a + b + c) &&- c \\
&3a + 3b + 3c
\end{aligned}$$

All of these, except for type $3a + 2b + c$, can be written as $3v$ or $3v + w$ for some Voronoi relevant vectors $v, w \in \mathcal{V}$. This means that they are, or are adjacent to, a domain centered at $3v$, and from Claim 2 it follows that they cannot intersect $\operatorname{dom}^{◖}(0, 3\Lambda)$. A similar argument shows that none of the domains at graph distance 4 from $\operatorname{dom}^{◖}(0, \Lambda)$ can intersect $\operatorname{dom}^{◖}(0, 3\Lambda)$, and therefore also none at higher graph distance.

Now the 51 offsets from $O_{\leq 2}$ are those that the argument from Lemma 3 yields, while O_3 contains the offsets corresponding to type $3a+2b+c$. As by Lemma 2 there are 57 intersecting domains, only 6 of the offsets from O_3 can correspond to intersecting domains. ◀