UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Delaunay triangulations of a family of symmetric hyperbolic surfaces in practice

THÈSE

présentée et soutenue publiquement le 12 mars 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Iordan Iordanov

Composition du jury

Rapporteurs :	Franz Aurenhammer	Graz University of Technology, Austria
	Stefan Schirra	University of Magdeburg, Germany
Examinateurs :	Guillaume Damiand	LIRIS - CNRS, Lyon, France
	Pierrick Gaudry	LORIA - CNRS, Nancy, France
	Mael Rouxel-Labbé	GeometryFactory, Sophia Antipolis, France
	Gert Vegter	University of Groningen, The Netherlands
Encadrante :	Monique Teillaud	LORIA - INRIA, Nancy, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Résumé

La surface de Bolza est la surface hyperbolique orientable compacte la plus symétrique de genre 2. Pour tout genre supérieur à 2, il existe une surface orientable compacte construite de manière similaire à la surface de Bolza et ayant le même type de symétries. Nous appelons ces surfaces des surfaces hyperboliques symétriques. Cette thèse porte sur le calcul des triangulations de Delaunay (TD) de surfaces hyperboliques symétriques.

Les TD de surfaces compactes peuvent être considérées comme des TD périodiques de leur revêtement universel (dans notre cas, le plan hyperbolique). Une TD est pour nous un complexe simplicial. Cependant, les ensembles de points ne définissent pas tous une décomposition simpliciale d'une surface hyperbolique symétrique. Dans la littérature, un algorithme a été proposé pour traiter ce problème avec l'utilisation de points factices : initialement une TD de la surface est construite avec un ensemble de points connu, puis des points d'entrée sont insérés avec le célèbre algorithme incrémental de Bowyer, et enfin les points factices sont supprimés, si la triangulation reste toujours un complexe simplicial. Pour la surface de Bolza, les points factices sont spécifiés. L'algorithme existant calcule une DT de la surface de Bolza comme une DT périodique du plan hyperbolique, ce qui nécessite de travailler dans un sous-ensemble approprié du plan hyperbolique.

Nous étudions les propriétés des TD de la surface de Bolza définies par des ensembles de points contenants l'ensemble proposé de points factices, et nous décrivons en détail une implémentation de l'algorithme incrémentiel pour cette surface. Nous commençons par définir un représentant canonique unique qui est contenu dans un sous-ensemble borné du plan hyperbolique pour chaque face d'une TD de la surface. Nous donnons une structure de données pour représenter une TD de la surface de Bolza via les représentants canoniques de ses faces. Nous détaillons les étapes de la construction d'une telle triangulation et les opérations supplémentaires qui permettent de localiser les points et de retirer des sommets. Nous présentons également les résultats sur le degré algébrique des prédicats nécessaires pour toutes les opérations.

Nous fournissons une implémentation entièrement dynamique pour la surface de Bolza, en offrant l'insertion de nouveaux points, la suppression des sommets existants, la localisation des points, et la construction d'objets duaux. Notre implémentation est basée sur la bibliothèque CGAL (Computational Geometry Algorithms Library), et est actuellement en cours de révision pour être intégrée dans la bibliothèque. L'intégration de notre code dans CGAL nécessite que tous les objets que nous introduisons soient compatibles avec le cadre existant et conformes aux standards adoptés par la bibliothèque. Nous donnons une description détaillée des classes utilisées pour représenter et traiter les triangulations hyperboliques périodiques et les objets associés. Des anal-

yses comparatives et des tests sont effectués pour évaluer notre implémentation, et une application simple est donnée sous la forme d'une démonstration CGAL.

Nous discutons une extension de notre implémentation à des surfaces hyperboliques symétriques de genre supérieur à 2. Nous proposons trois méthodes pour engendrer des ensembles de points factices pour chaque surface et présentons les avantages et les inconvénients de chaque méthode. Nous définissons un représentant canonique contenu dans un sous-ensemble borné du plan hyperbolique pour chaque face d'une TD de la surface. Nous décrivons une structure de données pour représenter une telle triangulation via les représentants canoniques de ses faces, et donnons des algorithmes pour l'initialisation de la triangulation. Enfin, nous discutons une implémentation préliminaire dans laquelle nous examinons les difficultés d'avoir des prédicats exacts efficaces pour la construction de TD de surfaces hyperboliques symétriques.

Mots-clés: Triangulation de Delaunay, géométrie hyperbolique, groupes Fuchsiens, CGAL.

Abstract

The Bolza surface is the most symmetric compact orientable hyperbolic surface of genus 2. For any genus higher than 2, there exists one compact orientable surface constructed in a similar way as the Bolza surface having the same kind of symmetry. We refer to this family of surfaces as symmetric hyperbolic surfaces. This thesis deals with the computation of Delaunay triangulations of symmetric hyperbolic surfaces.

Delaunay triangulations of compact surfaces can be seen as periodic Delaunay triangulations of their universal cover (in our case, the hyperbolic plane). A Delaunay triangulation is for us a simplicial complex. However, not all sets of points define a simplicial decomposition of a symmetric hyperbolic surface. In the literature, an algorithm has been proposed to deal with this issue by using so-called dummy points: initially a triangulation of the surface is constructed with a set of dummy points that defines a Delaunay triangulation of the surface, then input points are inserted with the well-known incremental algorithm by Bowyer, and finally the dummy points are removed, if the triangulation remains a simplicial complex after their removal. For the Bolza surface, the set of dummy points to initialize the triangulation is given. The existing algorithm computes a triangulation of the Bolza surface as a periodic triangulation of the hyperbolic plane and requires to identify a suitable subset of the hyperbolic plane in which to work.

We study the properties of Delaunay triangulations of the Bolza surface defined by sets of points containing the proposed set of dummy points, and we describe in detail an implementation of the incremental algorithm for it. We begin by identifying a subset of the hyperbolic plane that contains at least one representative for each face of a Delaunay triangulation of the surface, which enables us to define a unique canonical representative

in the hyperbolic plane for each face on the surface. We give a data structure to represent a Delaunay triangulation of the Bolza surface via the canonical representatives of its faces in the hyperbolic plane. We detail the construction of such a triangulation and additional operations that enable the location of points and the removal of vertices. We also report results on the algebraic degree of predicates needed for all operations.

We provide a fully dynamic implementation for the Bolza surface, supporting insertion of new points, removal of existing vertices, point location, and construction of dual objects. Our implementation is based on CGAL, the Computational Geometry Algorithms Library, and is currently under revision for integration in the library. To incorporate our code into CGAL, all the objects that we introduce must be compatible with the existing framework and comply with the standards adopted by the library. We give a detailed description of the classes used to represent and handle periodic hyperbolic triangulations and related objects. Benchmarks and tests are performed to evaluate our implementation, and a simple application is given in the form of a CGAL demo.

We discuss an extension of our implementation to symmetric hyperbolic surfaces of genus higher than 2. We propose three methods to generate sets of dummy points for each surface and present the advantages and shortcomings of each method. We identify a suitable subset of the hyperbolic plane that contains at least one representative for each face of a Delaunay triangulation of the surface, and we define a canonical representative in the hyperbolic plane for each face on the surface. We describe a data structure to represent such a triangulation via the canonical representatives of its faces, and give algorithms for the initialization of the triangulation with dummy points. Finally, we discuss a preliminary implementation in which we examine the difficulties of having efficient exact predicates for the construction of Delaunay triangulations of symmetric hyperbolic surfaces.

Keywords: Delaunay triangulations, hyperbolic geometry, Fuchsian groups, CGAL.

Acknowledgments

Acknowledgments (to be added in final version).

Contents

Résumé i

Abstract ii

Acknowledgments v

List of Figures xi

I Introduction 1

I.1 Motivation . . . 1

I.2 Problem statement . . . 2

I.3 Contribution . . . 3

II Background 5

II.1 Hyperbolic surfaces . . . 5

II.1.1 The hyperbolic plane . . . 5

II.1.2 Definition of compact hyperbolic surfaces . . . 8

II.1.3 Case study: the flat torus . . . 10

II.1.4 Curvature . . . 12

II.1.5 Curvature and universal cover . . . 14

II.1.6 Symmetric hyperbolic surfaces . . . 16

II.2 Triangulations . . . 20

II.2.1 Delaunay triangulations in the Euclidean plane . . . 21

II.2.2 Bowyer's incremental algorithm . . . 22

II.2.3 Dirichlet regions and Voronoi diagrams . . . 23

II.3 Triangulations of closed orientable surfaces . . . 23
II.3.1 Triangulations of the flat torus . . . 24
II.3.2 Triangulations of compact orientable hyperbolic surfaces . . . 27
II.3.3 Triangulations of the Bolza surface . . . 28

III Delaunay triangulations of the Bolza surface **31**
III.1 Representation of the triangulation . . . 31
III.1.1 Canonical representative of a face . . . 32
III.1.2 Data structure . . . 37
III.2 Construction of the triangulation . . . 39
III.2.1 Initialization . . . 39
III.2.2 Finding faces in conflict with a new point . . . 42
III.2.3 Insertion . . . 45
III.3 Operations on a triangulation . . . 47
III.3.1 Point location . . . 47
III.3.2 Removal of an existing vertex . . . 48
III.3.3 Removal of dummy points . . . 50
III.4 Algebraic complexity . . . 50
III.4.1 Predicates . . . 51
III.4.2 Algebraic degree of predicates . . . 53

IV Implementation in CGAL **59**
IV.1 A short introduction to CGAL . . . 59
IV.1.1 Predicates and Exact Geometric Computation . . . 60
IV.1.2 Techniques adopted by CGAL . . . 61
IV.2 Triangulations in CGAL . . . 63
IV.2.1 Historical notes . . . 63
IV.2.2 Declaration of a triangulation . . . 63
IV.2.3 Geometric traits . . . 65
IV.2.4 Kernels . . . 65
IV.2.5 Triangulation data structure . . . 66
IV.2.6 Faces and vertices . . . 66

IV.3 Periodic hyperbolic triangulations . . . 68
IV.3.1 Triangulation objects . . . 68
IV.3.2 Periodic hyperbolic geometric traits . . . 69
IV.3.3 Hyperbolic translations . . . 72
IV.3.4 Periodic hyperbolic vertices and faces . . . 76
IV.4 Application demo . . . 77
IV.5 Quantitative analysis of the source code . . . 81
IV.6 Experimental results . . . 82

V Delaunay triangulations of symmetric hyperbolic surfaces **87**
V.1 Representation of the triangulation . . . 87
V.1.1 Canonical representatives . . . 88
V.1.2 Data structure . . . 90
V.2 Computation of sets of dummy points . . . 91
V.2.1 The initial set – Weierstrass points . . . 92
V.2.2 From Γ_g to $\mathcal{N}_g$ . . . 97
V.2.3 Sequential strategy . . . 101
V.2.4 Sequential strategy with symmetries . . . 103
V.2.5 Deterministic strategy . . . 104
V.3 Construction of the triangulation . . . 106
V.3.1 Initialization . . . 107
V.3.2 Insertion and other operations . . . 115
V.4 Predicates . . . 115
V.5 Preliminary experimental results . . . 119
V.6 Conclusion . . . 120

VI Conclusions and future work **121**

Bibliography **125**

List of Figures

I.1 Salad and corals are negatively curved 2

II.1 Illustration of the fifth postulate of Euclid 6
II.2 Illustration of the Poincaré disk model for the hyperbolic plane 8
II.3 Distances in the Poincaré disk 8
II.4 Scheme of surface representations 10
II.5 Topological construction of a flat torus 11
II.6 Topological surfaces of genus 0, 1, and 2 12
II.7 Illustration of curvature of a smooth curve 13
II.8 Tiling of $\mathbb{H}^2$ with octagons and construction of 2-torus from an octagon . 16
II.9 Hyperbolic translations as side-pairing transformations 17
II.10 Illustration of a closed geodesic loop on the symmetric hyperbolic surface of genus 2 19
II.11 The generators of Γ_2 and the original domain for the Bolza surface 20
II.12 Comparison of Delaunay and non-Delaunay triangulation 22
II.13 Insertion with Bowyer's incremental algorithm 23
II.14 Covering spaces of the flat torus with 8 and 9 sheets 26
II.15 Comparison of Euclidean and hyperbolic Delaunay triangulation 28
II.16 Delaunay triangulation of the Bolza surface with dummy points 29

III.1 The original domain for the Bolza surface 32
III.2 Illustration of proof of Lemma III.1 33
III.3 Ordering and union of the neighboring Dirichlet regions 34
III.4 Illustration of proof of Theorem III.2 36
III.5 Illustration of candidate faces and canonical representatives 37
III.6 Initialization of a triangulation of $\mathbb{M}_2$ with the set $\mathcal{Q}_2$ 40
III.8 Triangulation of the Bolza surface with dummy points (screenshot) 44
III.9 Illustration of the neighbor translation 44
III.10 Illustration of the conflict zone (screenshot) 45
III.11 Illustration of point insertion (screenshot) 46
III.12 Point location in the hyperbolic plane 47

III.13 Illustration of the hyperbolic location (screenshot) . . . 48
III.14 Illustration of dummy point removal (screenshot) . . . 51
III.15 Illustration of the SideOfOriginalOctagon predicate . . . 52

IV.4 Illustration of a triangulation in CGAL . . . 66
IV.5 Vertex inheritance in CGAL . . . 67
IV.6 Face inheritance in CGAL . . . 67
IV.8 Periodic hyperbolic triangulation inheritance diagram . . . 70
IV.9 Geometric traits inheritance diagram . . . 72
IV.12 Vertex inheritance diagram . . . 76
IV.13 Face inheritance diagram . . . 77
IV.14 Screenshot of the demo window . . . 78
IV.15 Example of Delaunay triangulation and its dual on the Bolza surface . . . 79
IV.16 Free motion on the Bolza surface . . . 80
IV.17 Screenshots from the free motion demo . . . 80
IV.20 Histogram of the random points needed to remove all dummy points . . . 83
IV.21 Histogram of cardinality of minimal sets of points . . . 84
IV.22 Triangulation of the Bolza surface with 13 points . . . 85

V.1 Original domains for symmetric hyperbolic surfaces . . . 88
V.2 Intuitive illustration of Weierstrass points . . . 93
V.3 Illustration for hyperbolic trigonometry . . . 93
V.4 Structure of $DT_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$ . . . 96
V.5 Illustrations for the proof of Lemma V.13 . . . 98
V.6 Illustration for the proof of Proposition V.14 . . . 100
V.8 Sequential refinement strategy . . . 102
V.10 Sequential refinement strategy with symmetries . . . 104
V.12 Deterministic refinement strategy . . . 106
V.14 Zoom on the triangulation of the deterministic dummy points . . . 111
V.15 Initialization of triangulation with deterministic dummy points . . . 111
V.17 Evaluation of the SideOfOriginalPolygon predicate . . . 117
V.18 Triangulations with sequential dummy points . . . 119
V.19 Triangulations with symmetric dummy points . . . 119
V.20 Insertion of points for genus 2 and 3 . . . 120

VI.1 Triply periodic minimal surfaces . . . 123

Chapter I

Introduction

Delaunay triangulations in the Euclidean d-dimensional space $\mathbb{R}^d$ are among the most important structures in Computational Geometry and find applications in a vast array of domains of science and technology. They are well-studied and their mathematical properties are well understood. Many algorithms to compute them have been proposed, and the Computational Geometry Algorithms Library (CGAL [cga]) includes packages providing efficient implementations.

In this thesis, we consider the problem of computing Delaunay triangulations of a specific type of non-Euclidean spaces. More concretely, we work with a special family of compact, orientable, negatively curved surfaces. Triangulations of such surfaces are *periodic* and are important for many practical applications in physics, neuromathematics, material science, and other fields.

The terminology used in this chapter is explained in Chapter II.

I.1 Motivation

The motivation for this thesis is the rising need for periodic triangulations of negatively curved spaces. Consider the two objects shown in Figure I.1, a salad and a coral. Neither a salad leaf nor the surface of a coral can be laid flat on the Euclidean plane, but they can be flattened on the *hyperbolic plane*. Such objects are negatively curved, or *hyperbolic*, and we have already given two examples that exist in nature.

Other interesting objects with hyperbolic structure are, for example, the surfaces of the brain and the colon. Flattening them in the hyperbolic plane allows us to study them in a simpler and more efficient manner. Hyperbolic spaces are also used by physicists to study Einstein's theory of special relativity [Ung05].

Our goal in this thesis is to describe and implement an algorithm for the construction of Delaunay triangulations of hyperbolic spaces that are also periodic. The need to work in a periodic space may be dictated by an inherently periodic set of points, or because numerical simulations need to be run on spaces that are too large to be handled

Figure I.1: The leaves of salad (left) and the surface of corals (right) are hyperbolic. Their hyperbolic structure maximizes their surface area, which enables them to gather and process more nutrients.

by a modern computer: in such cases, scientists usually run experiments on a small sample that is repeated periodically to avoid boundary effects. Moreover, the study of triangulations of periodic spaces gives rise to interesting questions that do not arise for triangulations of non-periodic spaces, and thus provide important results.

Concrete examples that motivate our work on the computation of Delaunay triangulations of periodic hyperbolic spaces can be found in physics [STV08, BV86], neuromathematics [CFF11], and material science [ERH13a, ERH13b], to name a few.

I.2 Problem statement

The Bolza surface is arguably the most symmetric and easiest to construct compact orientable hyperbolic surfaces of genus 2. For any genus higher than 2, there exists one compact orientable surface constructed in a similar way as the Bolza surface, and it has the same kind of symmetry. We refer to this family of surfaces as *symmetric hyperbolic surfaces*. The problem that we undertake in this thesis is the following:

Given a set of points on a symmetric hyperbolic surface,
construct the Delaunay triangulation of the surface
defined by the given set of points.

In this thesis, a Delaunay triangulation is a simplicial complex. On a compact orientable surface, however, two points can be linked by multiple edges that do not cross each other, so it is quite easy to obtain triangles with loops and double edges. This is something we want to avoid, but there exist sets of points that cannot produce a simplicial decomposition of a symmetric hyperbolic surface. For example, in a decomposition with a single point, all edges are loops.

To deal with this problem, Bogdanov *et al.* [BTV16] proposed an algorithm for the triangulation of compact orientable surfaces based on the incremental algorithm by Bowyer [Bow81, Wat81]. The main problem that we tackle in this thesis is to implement the algorithm by Bogdanov *et al.* for symmetric hyperbolic surfaces of genus at least 2. A number of difficulties arise depending on the genus of the surface.

The algorithm proposed by Bogdanov *et al.* requires to construct initially a triangulation of the surface with a set of *dummy points* that is known to define a Delaunay triangulation of the surface. The set of input points are then added to this triangulation, and in the end the dummy points are removed, if the triangulation remains a simplicial complex after their removal.

For genus 2, the algorithm is specified in detail, up to the set of dummy points to be used in the initialization of the triangulation. The algorithm computes a periodic triangulation of the fundamental domain of the surface. For the construction of this triangulation, computations are required in a bounded subset of $\mathbb{H}^2$ that covers the fundamental domain of the surface. One of the things that is missing from the work of Bogdanov *et al.* is the description of such a bounded subset of $\mathbb{H}^2$ that contains a periodic triangulation of the fundamental domain of the surface with any set of points that contains the dummy points. Other missing details concern the formulation of all the operations on the triangulation (insertion, removal, location) while keeping track of its periodicity.

For higher genus, the first problem that must be solved is the choice of a set of dummy points, and their triangulation: for genus 2, since the set of dummy points is known, the combinatorics of their Delaunay triangulation is also known, but for higher genus we know neither a suitable set of points, nor its combinatorics. Next, as for genus 2, a bounded subset of $\mathbb{H}^2$ must be identified that contains a periodic triangulation of the fundamental domain of the surface with any set of points that contains the dummy points. Additionally, the computations for genus 2 involve algebraic numbers of degree 4, but for higher genus the computations involve algebraic numbers of higher degree, and it is a challenge to perform such computations exactly and efficiently.

Finally, we aim to include our implementation in CGAL, which adds to the difficulty of the implementation. Integrating our code into CGAL requires that all the objects that we introduce must be compatible with the existing framework and comply with the standards adopted by the library.

I.3 Contribution

Our contribution in this thesis is presented in three chapters.

In Chapter III we describe in detail an implementation of the algorithm proposed by Bogdanov *et al.* for genus 2. We identify a subset of $\mathbb{H}^2$ that contains a periodic Delaunay triangulation of the surface defined by sets of points containing the dummy points. We precise the steps of the construction of a periodic Delaunay triangulation of

the fundamental domain of the surface, and describe additional operations that allow to remove vertices from the triangulation. We also report results on the algebraic degree of predicates needed for all operations.

In Chapter IV we provide a fully dynamic implementation of the algorithm for the Bolza surface, supporting insertion of new points, removal of existing vertices, point location, and construction of dual objects. We give a detailed description of the classes used to represent and handle the triangulation and related objects. Benchmarks and tests are performed to provide interesting insight into the properties of Delaunay triangulations of the Bolza surface, and a simple application is given in the form of a CGAL demo.

In Chapter V we discuss a generalization of the ideas presented for genus 2 to symmetric hyperbolic surfaces of higher genus. We discuss three methods to generate sets of dummy points for each surface and discuss their advantages and shortcomings. We identify a suitable subset of $\mathbb{H}^2$ that contains a periodic triangulation of the fundamental domain of a symmetric hyperbolic surface of genus g defined by a set of points containing the set of dummy points. Finally, we present a preliminary implementation of our generalized algorithm in which we examine the difficulties of efficient exact computations for the construction of Delaunay triangulations of symmetric hyperbolic surfaces.

Chapter II

Background

In this chapter we describe the notions that are in the very basis of this thesis. We start our discussion with hyperbolic surfaces, the way in which we represent them in this thesis, and their properties. We continue with Delaunay triangulations, their definition on closed compact surfaces, and their properties. We conclude this chapter with a discussion about existing work on the construction of Delaunay triangulations of closed compact surfaces. The familiar reader may skip over any of these sections.

II.1 Hyperbolic surfaces

In this section we present the objects with which we work and the space in which they live. We begin by introducing the hyperbolic plane and hyperbolic surfaces. To give an analogy with a more familiar Euclidean surface, we discuss some of the properties of the flat torus that are in direct correspondence with the properties of compact hyperbolic surfaces. We continue with well-known fundamental results that connect the geometry and topology of hyperbolic surfaces, and we conclude with the introduction of *symmetric* hyperbolic surfaces, the central objects of this thesis.

II.1.1 The hyperbolic plane

The hyperbolic plane is the space in which we work in this thesis, and in this section we present it and its properties. We start by discussing its conception.

The history of hyperbolic geometry

Geometry is historically among the first branches of mathematics to be established as a science. In the beginning it consisted of empirical rules used mainly to measure and divide plots of land, hence the name "γεωμετρία" (γέω = earth, μέτρον = measure). The Greek philosopher Euclid of Alexandria was the first to formalize the basic rules according to which geometry works. His most famous work is a treatise consisting of

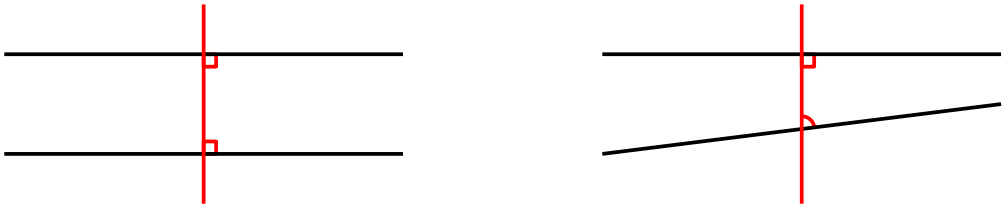

Figure II.1: Illustration of the fifth postulate of Euclid. Given two lines and a third one incident to both, if the sum of the internal angles on one side is two right angles (left), then the lines are parallel. If the sum of the internal angles on one side is less than two right angles (right), then the lines intersect on that side.

13 books, the *Elements* (Στοιχεία) [Euc]. In the beginning of the first book, he gives five rules, called *postulates*, that set the whole basis for what we call today Euclidean geometry. These rules represented the world around him as he could observe it, and are reported in their original form below.

a) Ἠιτήσθω ἀπὸ παντὸς σημεῖου ἐπὶ πᾶν σημεῖον εὐθεῖαν γραμμὴν ἀγαγεῖν.
b) Καὶ πεπερασμένην εὐθεῖαν κατὰ τὸ συνεχὲς ἐπ' εὐθεῖας ἐκβαλεῖν.
c) Καὶ παντὶ κέντρῳ καὶ διαστήματι κύκλον γράφεσθαι.
d) Καὶ πάσας τὰς ὀρθὰς γωνίας ἴσας ἀλλήλαις εἶναι.
e) Καὶ ἐὰν εἰς δύο εὐθείας εὐθεῖα ἐμπίπτουσα τὰς ἐντὸς καὶ ἐπὶ τὰ αὐτὰ μέρη γωνίας δὺο ὀρθῶν ἐλάσσονας ποιῇ, ἐκβαλλομένας τὰς δύο εὐθείας ἐπ' ἄπειρον συμπίπτειν, ἐφ' ἃ μέρη εἰσὶν αἱ τῶν δύο ὀρθῶν ἐλάσσονες.

Euclid's five postulates are usually interpreted in English, their meaning is given rather than the actual statement. Here I attempt to give a direct translation as faithful as possible to the original statements, with my current level of knowledge of Ancient Greek:

a) Let it be required that from any point to any point a straight line may be drawn.
b) And also that a finite line may continuously extend onto an infinite line.
c) And also that with any center and any distance a circle may be drawn.
d) And also that all right angles are equal among themselves.
e) And also that if a line incident to two lines creates two internal angles on the same side that measure less than two right angles, extended to infinity the two lines intersect, on the side on which the two angles measure less than two right.

The fifth postulate (both in the original text and in translation) is quite complicated. It becomes clearer with the illustration shown in Figure II.1, which shows an alternative way to state it: *given a line and a point not on it, there is a* ***unique*** *line through the point parallel to the given line.* Even with explanation, however, this postulate is unnaturally more complicated than the other four. Many mathemati-

cians believed that in fact the first four postulates can be used to derive the fifth. After many failed attempts to do so, finally around 1830 János Bolyai and Nikolai Lobachevski independently published their treatises on *hyperbolic geometry*.

Hyperbolic geometry, like Euclidean geometry, also has five postulates. The first four are the same, but the fifth is different: *given a line and a point not on it, there exist* ***infinitely many*** *lines through the point parallel to the given line.* The hyperbolic plane $\mathbb{H}^2$ is a two-dimensional space that obeys the five postulates of hyperbolic geometry. One of the properties of the hyperbolic plane is that its Gaussian curvature at each point is equal to -1. We will discuss curvature in Section II.1.4. The hyperbolic plane has a *metric structure*, which means that we can measure distances and angles on it. However, Hilbert proved in 1901 [Hil01] that there is no C^2-smooth embedding of $\mathbb{H}^2$ in $\mathbb{R}^3$.

There exist several *models* to represent $\mathbb{H}^2$ [Rat06, Chapter 1], where each model is a different projection of $\mathbb{H}^2$ onto $\mathbb{R}^2$ (e.g., the Beltrami-Klein model, or the Poincaré disk and half-plane models) or $\mathbb{R}^3$ (e.g., the hyperboloid model). Each model preserves certain features of $\mathbb{H}^2$ and distorts others. In this thesis, we use the Poincaré disk model.

The Poincaré disk model

In the Poincaré disk model, the whole hyperbolic plane is represented as the open unit disk centered at the origin of the Euclidean plane. Points on the unit circle are points at infinity. This model is *conformal*, which means that angles are measured as in the Euclidean setting. It distorts distances, so objects of the same size and shape appear exponentially smaller as we get closer to the boundary. Hyperbolic lines (or geodesics) are represented as open diameters of the unit disk, and as open Euclidean circular arcs perpendicular to its boundary. Hyperbolic circles coincide with Euclidean circles contained in the unit disk, although a circle contained in the unit disk has different Euclidean and hyperbolic centers and radii. See Figure II.2.

The Poincaré disk model appears also in art, notably in the works of M.C. Escher. In his works Escher captures the distance distortion in this model by illustrating shapes of the same shape and size that tile $\mathbb{H}^2$. See Figure II.2 - Right for one of Escher's works.

Distances in the Poincaré disk model depend on the location of points. We denote with $d_{\mathbb{E}}(\cdot,\cdot)$ the Euclidean distance, and with $d_{\mathbb{H}}(\cdot,\cdot)$ the hyperbolic distance. The hyperbolic distance of a point r from the origin O is given as

$$d_{\mathbb{H}}(O,r) = \log\left(\frac{1+d_{\mathbb{E}}(O,r)}{1-d_{\mathbb{E}}(O,r)}\right),$$

where $\log(\cdot)$ is the natural logarithm.

Any two distinct points p and q in $\mathbb{H}^2$ define a unique hyperbolic line that meets the boundary of the unit disk at two points a and b, as shown in Figure II.3. The hyperbolic

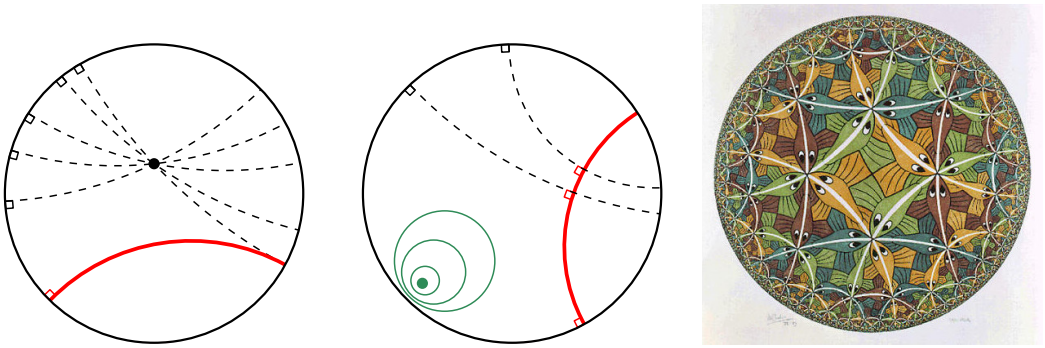

Figure II.2: ***Left:*** *Parallel lines in the Poincaré disk model. All the dashed lines passing through the solid black point are parallel to the thick red line.* ***Center:*** *The two dashed hyperbolic lines are both orthogonal to the solid red hyperbolic line; note that the angle is right in the Euclidean sense. The three circles have the same center (the solid green point).* ***Right:*** *The work* "Circle Limit III" *by M.C. Escher illustrates the metric deformation in the Poincaré disk: all fish are the same size and shape. All M.C. Escher works* © *2018 The M.C. Escher Company - the Netherlands. All rights reserved. Used by permission.* www.mcescher.com

distance of the points p and q is given as

$$d_{\mathbb{H}}(p,q) = \log\left(\frac{d_{\mathbb{E}}(a,q)\cdot d_{\mathbb{E}}(p,b)}{d_{\mathbb{E}}(a,p)\cdot d_{\mathbb{E}}(q,b)}\right).$$

II.1.2 Definition of compact hyperbolic surfaces

A surface is a 2-manifold. In this thesis we work with a very specific family of compact orientable hyperbolic surfaces that we discuss in detail in Section II.1.6. A hyperbolic surface is a surface that is locally "similar" to $\mathbb{H}^2$. A more formal definition is the following.

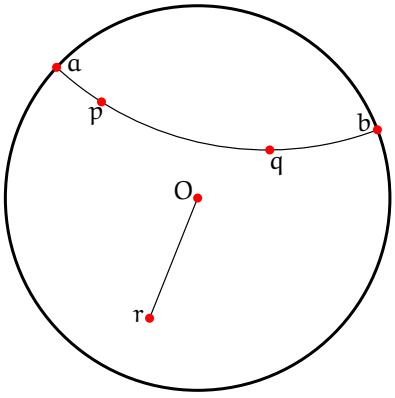

Figure II.3: Distances in the Poincaré disk depend on the location of points.

Definition II.1 (Hyperbolic surface)**.** A *hyperbolic surface* $\mathbb{M}$ is a compact topological space together with a covering by finitely many open sets $\mathbb{M} \subseteq \bigcup_{i=0}^{n} O_i$ such that $\forall\, i\ \exists\, f_i : O_i \to U_i \subset \mathbb{H}^2$ with the properties:

- $\forall\, i$ f_i is a homeomorphism;
- $\forall\, i, j$ $\left(f_j\big|_{O_i \cap O_j}\right) \circ f_i^{-1} : U_i \to U_j$ is a Möbius transformation.

Each open set O_i and its transition function f_i define a local *map* of $\mathbb{M}$. The set of all maps of $\mathbb{M}$ fully describes the surface, and is called an *atlas* of $\mathbb{M}$.

By Definition II.1, we only work with *smooth* surfaces, since Möbius transformations are differentiable. Moreover, we consider only surfaces for which the Jacobian of each transition map f_i in the atlas of $\mathbb{M}$ has positive determinant. Such surfaces are *orientable*; intuitively, an orientable surface has two distinct *sides*.

The transition maps f_i transport the metric structure of the hyperbolic plane onto $\mathbb{M}$. The distance between two points on $\mathbb{M}$ is measured by finding the images of the two points in $\mathbb{H}^2$ via the transition maps, considering all paths from one point to the other through overlapping open sets in the atlas of $\mathbb{M}$, and finding the shortest among these paths according to the distance defined in $\mathbb{H}^2$.

Before we proceed further, we make a convention regarding the figures in this thesis. As we mentioned, hyperbolic surfaces cannot be smoothly embedded in $\mathbb{R}^3$, so we need to resort to different representations that capture interesting properties of a surface. A schema of the different representation we will use, as well as the ways in which we transition from one to the other, are shown in Figure II.4.

We will often resort to a representation that captures the topology of a surface via its genus. This topological representation is easily drawn in $\mathbb{R}^3$ and it enables us to show the smoothness of the surface. One of its shortcomings is that we are unable to show accurately distances and angles on the surface.

Another representation that we will use shows a surface as a polygon, called *fundamental domain*, that we discuss in Section II.1.6. With this representation, a surface is given as a polygon together with a set of rules to glue its sides together. Gluing the sides of the polygon according to the rules gives the surface. This representation enables us to accurately depict angles and distances, and different polygons represent different surfaces. We will work only with a specific family of hyperbolic surfaces whose fundamental domain is a *regular* polygon, and with gluing rules such that we only glue opposite sides together. The downside of this representation is that the topology and the smoothness of the surfaces are not evident, although they can be inferred by the Gauss-Bonnet theorem, which we will discuss in Section II.1.5.

Lastly, a hyperbolic surface is tightly connected to its *universal cover*, which, for the surfaces in which we are interested, is the hyperbolic plane $\mathbb{H}^2$. We will see more details in Sections II.1.3 and II.1.6.

Throughout the thesis, we will often switch between the topological representation

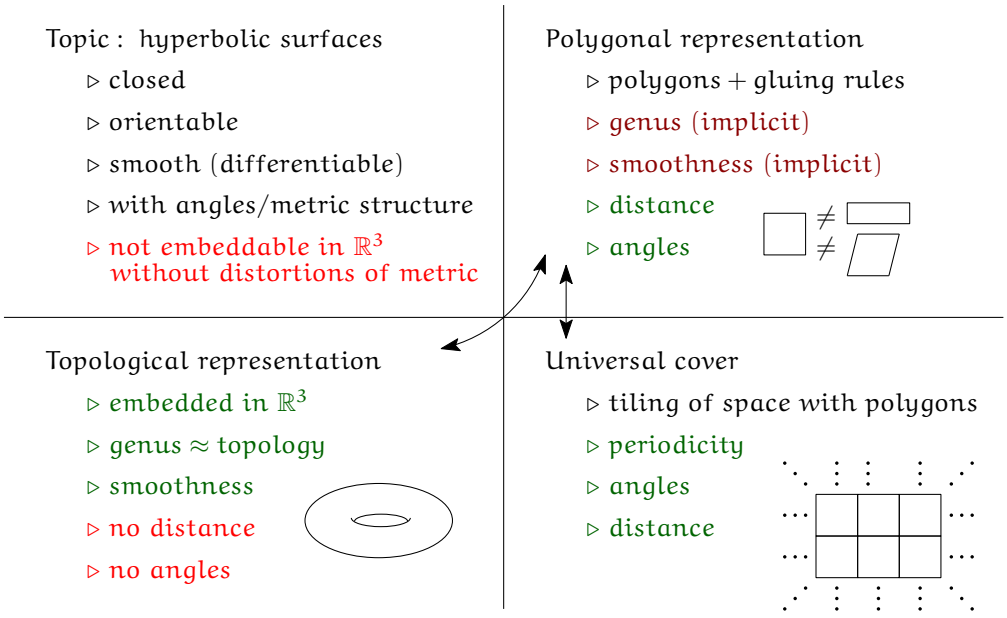

Figure II.4: Scheme of surface representations in this thesis. The arrows indicate transitions between different representations that we will use in this thesis.

of a surface and its fundamental domain. We will also often go back and forth between its fundamental domain and universal cover.

II.1.3 Case study: the flat torus

In this section we discuss a Euclidean compact orientable surface whose construction and properties are in direct analogy with the construction and properties of hyperbolic compact orientable surfaces.

Consider the square shown in the top left part of Figure II.5. We assign directions to its opposite sides and glue them together so that the directions agree.[1] The result is a Euclidean compact orientable surface that is called a *flat torus*. A flat torus cannot be embedded in $\mathbb{R}^3$ without distorting its metric structure due to a classical result by Hartman and Nirenberg [HN59], so the construction shown in Figure II.5 is only topological. A relatively new and very interesting work describes and shows a C^1-smooth embedding of a flat torus in $\mathbb{R}^3$ [BJLT12].

The whole Euclidean plane $\mathbb{E}^2$ can be covered with translated copies of the initial square. To get a sense of the orientation of the square, we place an ant on it, as shown in the bottom row of Figure II.5. Instead of gluing the sides of the square together, we copy and translate the square on the plane, as shown in the figure, without rotating or reflecting it. First, notice that the union of finitely many translated copies of the square

[1] Animation of the gluing process that gives a flat torus: https://tinyurl.com/torus-gluing

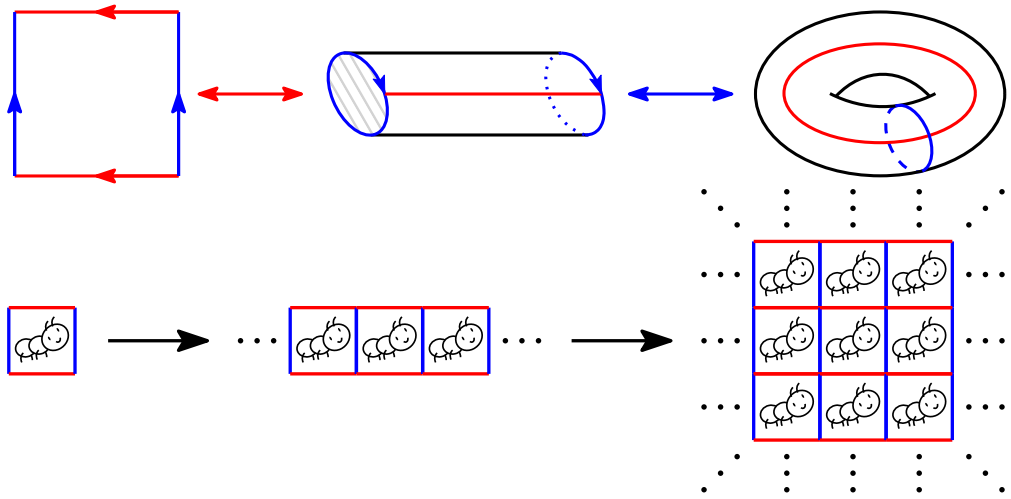

Figure II.5: ***Top:*** *topological construction of a flat torus.* ***Bottom:*** *tiling of the universal cover of the flat torus with copies of its fundamental domain. Note that only translations are used.*

is contained in $\mathbb{E}^2$ and "covers" the initial square. Second, if we consider infinitely many copies of the square, we can cover the whole Euclidean plane. With these two facts, $\mathbb{E}^2$ is the *universal cover* of the square, and by extension of the surface $\mathbb{M}$ that it defines. The square defines a *periodic* tiling of the Euclidean plane, and the ant shows the pattern of the periodicity from square to square.

The flat torus, being a Euclidean surface, has a definition similar to Definition II.1, with the difference that its transition functions map open sets on the torus to open sets in $\mathbb{E}^2$, so its atlas lives in $\mathbb{E}^2$. Therefore, distances on the torus are measured with the Euclidean metric that is transported from the Euclidean plane to the surface via the transition functions. A *geodesic curve* is a curve that locally minimizes distances; it can be viewed as the generalization of a straight line in the Euclidean plane to other spaces. A geodesic on the flat torus is the image via the transition functions of a geodesic (i.e., segment) on the Euclidean plane.

An interesting property of the flat torus is that between two points on the surface there exist multiple geodesic curves. Moreover, if we consider just a single point, there are multiple geodesic *loops* from the point to itself that cannot be continuously reduced to a single point. Such geodesic loops are called *nontrivial*. Consider, for instance, the square shown in the upper left part of Figure II.5. When the opposite sides of the square are glued, its four corners meet in a single point. The blue and red sides of the square are Euclidean segments, so with the gluing they become nontrivial geodesic loops on the flat torus from the corner point to itself.

A very important property of compact surfaces is their *systole*.

Definition II.2 (Systole)**.** The length of the shortest nontrivial closed geodesic curves

on a compact surface $\mathbb{M}$ is called the *systole* of $\mathbb{M}$, denoted as $\text{sys}(\mathbb{M})$.

For the flat torus constructed from a square, the systole is equal to the side length of the square.

We discuss now the topology of the torus. For a surface $\mathbb{M}$, the number of handles needed to attach to a sphere to obtain a surface topologically equivalent (i.e., homeomorphic) to $\mathbb{M}$ is the *genus* of $\mathbb{M}$. The torus is homeomorphic to a sphere with one handle attached, so it is a surface of genus 1. See Figure II.6.

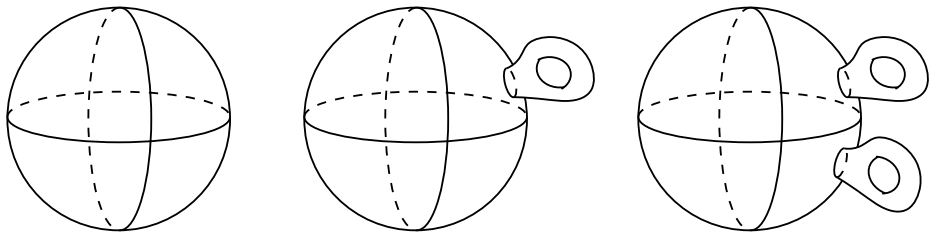

Figure II.6: Surfaces of genus 0, 1, and 2. The genus of a surface $\mathbb{M}$ is the number of handles one needs to attach to a sphere in order to obtain a surface topologically equivalent to $\mathbb{M}$.

The genus of a surface $\mathbb{M}$ is strongly related to the *Euler characteristic* $\chi(\mathbb{M})$ of the surface. The Euler characteristic is a topological invariant, i.e., it is a quantity that does not change no matter how the surface is bent or twisted, as long as its topology remains the same. The relationship between the genus g of a surface $\mathbb{M}$ and its Euler characteristic is the following:

$$\chi(\mathbb{M}) = 2(1 - g). \tag{II.3}$$

It is interesting to note that for $g = 0$, the Euler characteristic is positive, for $g = 1$ it is zero, and for $g \geq 2$ it is negative. In this section we discussed the properties of the flat torus, a surface of genus 1, as a case study, but the thesis treats triangulations of surfaces of genus at least 2. The fact that the Euler characteristic of surfaces of genus at least 2 is negative implies that such surfaces share some common features that are not found in the flat torus. To gain better insight into these features, in the next section we discuss curvature, which will enable us to establish a connection between geometry and topology.

II.1.4 Curvature

In this section we define curvature via *osculating circles* [ASG06, Section 4.4]. Let $\mathbb{L}$ be a smooth curve, and let p be a point on $\mathbb{L}$. Let p_1 and p_2 be two points on $\mathbb{L}$ on both sides of p. Consider the circle passing through p_1, p and p_2. The osculating circle at p is obtained by taking the limits $p_1 \to p$ and $p_2 \to p$. The curvature of $\mathbb{L}$ at p is $\kappa(p) = \frac{1}{r_o}$, where r_o is the radius of the osculating circle at p. By taking into account

also on which side of $\mathbb{L}$ is the osculating circle, we obtain the *signed* curvature at p. See Figure II.7 for an illustration.

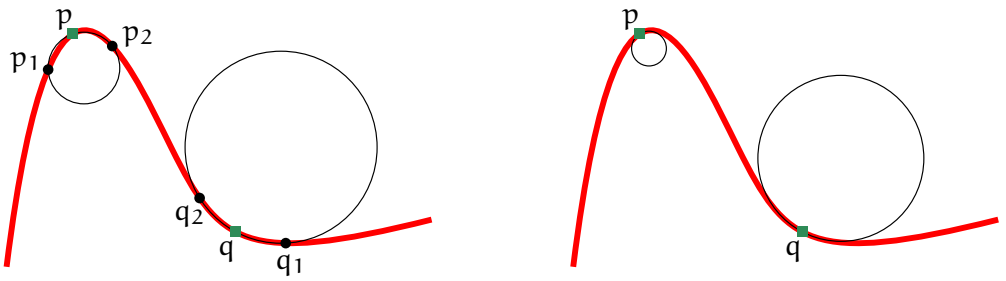

*Figure II.7: Illustration of curvature of a smooth curve in two points p and q. **Left:** For any point p on the curve $\mathbb{L}$, consider two other points p_1, p_2 on $\mathbb{L}$ on either side of p. Take the circle passing through the three points. **Right:** Let the points p_1, p_2 approach p. At the limit we obtain the osculating circle at p. The curvature of the osculating circle at p is the curvature of $\mathbb{L}$ at p.*

The curvature of a smooth surface $\mathbb{M}$ at a point p is defined in a similar manner. A plane containing the normal vector at a point p of $\mathbb{M}$ is called a *normal plane*, and the intersection of a normal plane and $\mathbb{M}$ is called a *normal section*. Each normal section is a smooth curve, for which the signed curvature at p is defined as above. By taking the maximum and minimum over the curvatures defined by all the normal planes through p, we obtain the two *principal curvatures* at p, denoted as $\kappa_1(p)$ and $\kappa_2(p)$, respectively. The *Gaussian curvature* at p is the product of the two principal curvatures:

$$\kappa(p) = \kappa_1(p) \cdot \kappa_2(p).$$

The curvature and the genus of a surface, a geometric and a topological characteristic respectively, are connected via the Gauss-Bonnet theorem.

Theorem II.4 (Gauss-Bonnet theorem [AW43]). *Let $\mathbb{M}$ be a smooth compact orientable surface of genus g with no boundary and Gaussian curvature κ at each point. The curvature and the genus of $\mathbb{M}$ are related by*

$$\int_{\mathbb{M}} \kappa \, dA = 2\pi \, \chi(\mathbb{M}), \tag{II.5}$$

where dA is an element of area of the surface, and $\chi(\mathbb{M})$ is the Euler characteristic of $\mathbb{M}$.

Note that the integral $\int_{\mathbb{M}} \kappa \, dA$ is the Gaussian curvature of the surface $\mathbb{M}$. From (II.3), we get

$$\int_{\mathbb{M}} \kappa \, dA = 4\pi(1 - g). \tag{II.6}$$

For $g = 1$, the right-hand side of (II.6) becomes 0, which implies that the Gaussian curvature of the flat torus is 0. For $g \geq 2$, the right-hand side is negative, implying that the Gaussian curvature of compact orientable surfaces of genus at least 2 is negative.

As we discussed in Section II.1.3, a flat torus can be constructed from a square in the Euclidean plane. A reasonable question to ask is, can surfaces of genus 2 also be constructed from Euclidean polygons? The answer comes from the *Uniformization theorem*, conjectured first by Felix Klein and Henri Poincaré, and later proved by Poincaré and Paul Koebe.

Theorem II.7 (Uniformization theorem). *The universal cover of any closed orientable surface with constant Gaussian curvature* k *is either*

- *the sphere, if* $k > 0$,
- *the Euclidean plane, if* $k = 0$,
- *or the hyperbolic plane, if* $k < 0$.

II.1.5 Curvature and universal cover

From the Gauss-Bonnet theorem and from the Uniformization theorem, no surface of genus higher than 1 can be constructed by a Euclidean polygon. Such surfaces can be constructed only by polygons in the hyperbolic plane. A polygon in $\mathbb{H}^2$ is a bounded convex subset of $\mathbb{H}^2$ delimited by finitely many geodesic segments. A subset of $\mathbb{H}^2$ is convex if the geodesic segment between any two points in the set is contained in the set. Particularly for hyperbolic triangles, the Gauss-Bonnet formula has the following special form.

Theorem II.8 (Gauss-Bonnet theorem for hyperbolic triangles [AW43]). *Let* σ *be a triangle in* $\mathbb{H}^2$ *with angles* $\vartheta_1, \vartheta_2, \vartheta_3$. *Then the area of* σ *is given by*

$$\mathit{Area}(\sigma) = \pi - \sum_{k=1}^{3} \vartheta_k.$$

By taking a polygon in $\mathbb{H}^2$ and subdividing it into triangles, we obtain a generalization of Theorem II.8 to polygons.

Theorem II.9 (Gauss-Bonnet theorem for hyperbolic polygons). *Let* P *be a hyperbolic polygon with* n *sides, and internal angles* $\vartheta_1, \vartheta_2, \ldots, \vartheta_n$. *Then the area of* P *is*

$$\mathit{Area}(P) = (n-2)\pi - \sum_{k=1}^{n} \vartheta_k.$$

We give now more details about why a hyperbolic surface cannot be constructed by a Euclidean polygon. Recall the definition for the Euler characteristic based on the genus that we gave in (II.3). In its classical form, the Euler characteristic χ is defined for the surfaces of polyhedra. The fact that it is a topological invariant makes it possible for the definition to be extended to smooth surfaces, since they can be "polygonized" (e.g., triangulated) without changing their topology.

Let $\mathbb{M}$ be a smooth, closed, orientable surface of genus $g \geq 2$. For a polygonized version of $\mathbb{M}$ with V vertices, F faces and E edges, the Euler characteristic is given as

$$\chi(\mathbb{M}) = V - E + F. \tag{II.10}$$

By combining the two expressions for the Euler characteristic (II.3) and (II.10), we get

$$V - E + F = 2(1 - g). \tag{II.11}$$

Suppose now that $\mathbb{M}$ is polygonized in a very simple way, by cutting along lines that start and end at the same point on the surface and requiring that the result is a polygon D in $\mathbb{H}^2$. We consider one point p on the surface $\mathbb{M}$, so $V = 1$. We require that the end result is a polygon, therefore $F = 1$. Substituting these numbers into (II.11) yields

$$V - E + F = 2(1 - g) \Rightarrow 2 - E = 2 - 2g \Rightarrow E = 2g.$$

This result implies that we need to cut along $2g$ closed curves through p in order to obtain a decomposition of $\mathbb{M}$ with one vertex and one face. Since every curve along which we cut is the gluing of two sides of D, the number of sides of D is double. So, the fundamental domain D of $\mathbb{M}$ has $4g$ sides. Note also that, by construction, all vertices of the polygon D are images of the same point p on the surface where all the cuts meet.

So, a surface of genus 2 is constructed by an octagon. If the Euclidean plane were to be the universal cover for a surface of genus 2, we should be able to tile the Euclidean plane with octagons, which is not possible as its dual would be a planar graph with average degree 8, in contradiction with Euler's characteristic. It is possible, however, to tile the hyperbolic plane with octagons. Figure II.8 shows a tiling of the hyperbolic plane with regular octagons and the topological construction of a surface of genus 2 from a regular hyperbolic octagon.

We can go one step further with (II.9) to get a very interesting result. We started with the assumption that $\mathbb{M}$ is a smooth surface, therefore the sum of the internal angles of D is 2π. Moreover, we know that D is a polygon with $4g$ angles, so $n = 4g$. Therefore,

$$\text{Area}(D) = (4g - 2)\pi - 2\pi = 4\pi(g - 1). \tag{II.12}$$

Since the area of D is equal to the area of $\mathbb{M}$, and $\mathbb{M}$ is any smooth, compact, orientable surface, Equation (II.12) connects the *area* of $\mathbb{M}$, a purely geometric property, to its *genus*, a purely topological property. The implication of this result is that

the area of any smooth, compact, orientable surface of genus $g \geq 2$ is equal to $4\pi(g - 1)$.

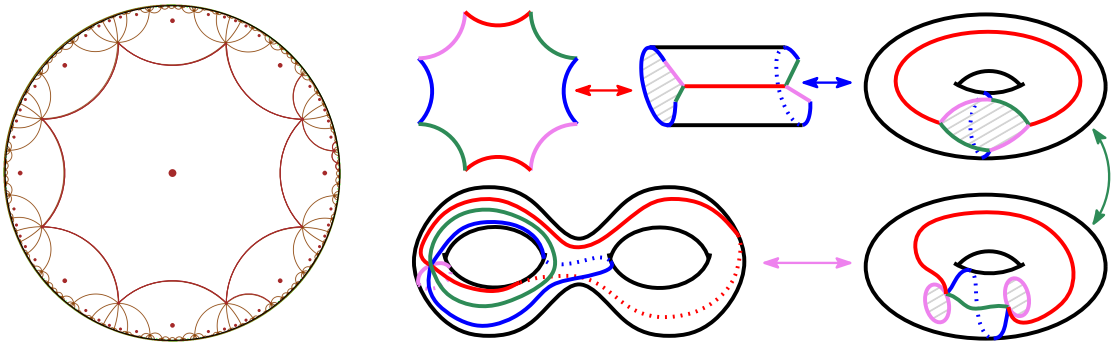

Figure II.8: ***Left:*** *Tiling of the hyperbolic plane with regular hyperbolic octagons. All the angles of each octagon are equal to* $\frac{\pi}{4}$. ***Right:*** *Construction of a surface of genus 2 from an octagon. The colored arrows show which sides are glued together to go to the next step. Note that the process is reversible: we can cut along the curves embedded in the 2-torus to obtain an octagon.*

II.1.6 Symmetric hyperbolic surfaces

In this section we concentrate on the specific family of surfaces with which we work in this thesis. To precisely define these surfaces, we need to introduce hyperbolic translations and Fuchsian groups. For more details on these topics, see [Kat92].

Hyperbolic translations

Hyperbolic translations are mappings that act on $\mathbb{H}^2$ and preserve both distances and orientation. A hyperbolic translation is a *Möbius transformation* and is given in the form

$$T(z) = \frac{\alpha \cdot z + \beta}{\overline{\beta} \cdot z + \overline{\alpha}}, \quad z \in \mathbb{H}^2, \quad \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 - |\beta|^2 = 1, \tag{II.13}$$

where $\overline{\alpha}$ and $\overline{\beta}$ are the complex conjugates of α and β, respectively. Note that points in the Poincaré disk are considered here as complex numbers in the open unit disk. Each hyperbolic translation can be represented with a 2×2 complex matrix

$$T = \begin{bmatrix} \alpha & \beta \\ \overline{\beta} & \overline{\alpha} \end{bmatrix}, \qquad \alpha, \beta \in \mathbb{C}, \quad \det(T) = 1. \tag{II.14}$$

A key difference between Euclidean and hyperbolic translations is that hyperbolic translations do not commute. This property is implied by the fact that the combination of two hyperbolic translations amounts to the multiplication of their matrices, which is generally non-commutative.

Each hyperbolic translation T fixes two points at infinity (i.e., on the unit circle) that remain invariant under the action of the translation, and no point in $\mathbb{H}^2$. The hyperbolic line through the two fixed points of T is called the *axis of* T $\mathrm{Ax}(T)$. Applying T to a

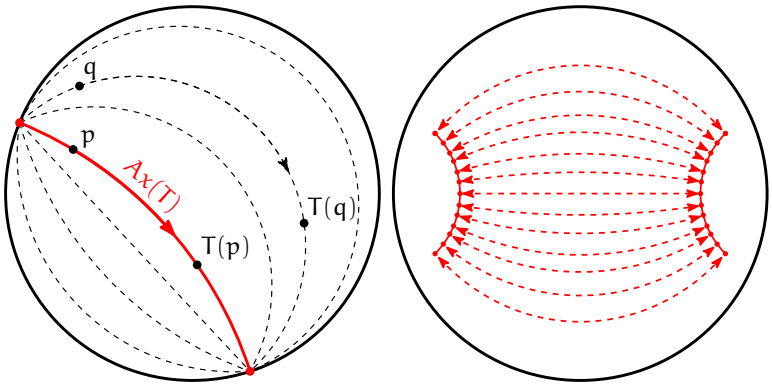

Figure II.9: ***Left:*** *Illustration of a hyperbolic translation* g*. The solid red curve is the axis of* g*, and* $d_{\mathbb{H}}(p, T(p)) = \ell(T)$*. All dashed curves are* ***not geodesics****: each of them is a locus of points equidistant from* $\mathrm{Ax}(T)$*. Any point* q *not on the axis of* T *is moved along such a curve, and* $d_{\mathbb{H}}(q, T(q)) > \ell(T)$*.* ***Right:*** *A hyperbolic translation applied to a geodesic segment produces another geodesic segment. Hyperbolic translations encode the gluing rules that produce a compact hyperbolic surface.*

point that lies on $\mathrm{Ax}(T)$ moves the point along $\mathrm{Ax}(T)$ by a distance that is constant and is called the *translation length* $\ell(T)$ of T. The translation length of T is given by the trace of its matrix as

$$\ell(T) = 2 \cdot \operatorname{arcosh}\left(\tfrac{1}{2}\operatorname{Tr}(T)\right).$$

Points that do not lie on $\mathrm{Ax}(T)$ are moved along curves that are loci of points equidistant from $\mathrm{Ax}(T)$. Such points are moved by a distance that is at least equal to the translation length (but is generally larger). See Figure II.9 - Left for an illustration.

Hyperbolic translations encode mathematically the gluing rules that we mentioned in Section II.1.2, so they are called *side-pairing transformations*. See Figure II.9 - Right for an illustration of how two sides of the regular octagon are "glued together" by a hyperbolic translation.

Compact orientable surfaces as quotient spaces

Let D_g be the regular polygon with angle sum 2π centered at the origin of $\mathbb{H}^2$, with $4g$ sides and $4g$ angles of equal measure. The fact that all the angles of D_g are equal implies that each one of its angles is equal to $\frac{2\pi}{4g} = \frac{\pi}{2g}$.

Let $T_0, T_1, \ldots, T_{4g-1}$ be the $4g$ hyperbolic translations that identify opposite sides of D_g so that $T_k = T_{k+2g}^{-1}$, where indices are taken modulo $4g$ and T^{-1} indicates the inverse of T. The translations T_k generate a group of hyperbolic translations (also known as a *Fuchsian group*) Γ_g with the following *finite presentation*:

$$\Gamma_g = \langle\, T_0, T_1, \ldots, T_{2g-1} \mid T_0 T_1^{-1} \ldots T_{2g-2} T_{2g-1}^{-1} T_0^{-1} T_1 \ldots T_{2g-2}^{-1} T_{2g-1} \,\rangle. \tag{II.15}$$

The notation in (II.15) means that each element of Γ_g can be written as a combination of the *generators* $T_0, T_1, \ldots, T_{2g-1}$ or their inverses, while the *relation* of Γ_g, the element $T_0 T_1^{-1} \ldots T_{2g-2} T_{2g-1}^{-1} T_0^{-1} T_1 \ldots T_{2g-2}^{-1} T_{2g-1}$, is equivalent to the identity translation of Γ_g.

Each point $p \in \mathbb{H}^2$ has infinitely many images under the action of Γ_g; the set of all these images is called the *orbit* of p under Γ_g and we denote it as $\Gamma_g p$. The polygon D_g contains at least one point from each orbit of Γ_g, therefore it is a *fundamental domain* for the action of Γ_g on $\mathbb{H}^2$. Note that on the boundary of D_g there are points in the same orbit. In particular, the vertices of D_g all belong in the same orbit.

So, each point in $\mathbb{H}^2$ belongs to an *equivalence class* under the action of Γ_g. The set of all these equivalence classes is a *quotient space*.

Definition II.16 (Symmetric hyperbolic surface of genus $g \geq 2$)**.** Let D_g be the regular hyperbolic polygon with angle sum 2π centered at the origin of $\mathbb{H}^2$, and let Γ_g be the Fuchsian group whose generators identify opposite sides of D_g. The *symmetric hyperbolic surface of genus* $g \geq 2$ is the quotient space

$$\mathbb{M}_g = \mathbb{H}^2 / \Gamma_g$$

together with the natural projection mapping $\pi_g : \mathbb{H}^2 \to \mathbb{M}_g$.

The group Γ_g is the *fundamental group* for the surface $\mathbb{M}_g$, and the polygon D_g is by extension a fundamental domain for the surface as well.

The matrices of the generators of $\mathbb{M}_g$ for $k = 0, 1, \ldots, 4g-1$ are given as

$$T_k = \begin{bmatrix} \cot\left(\frac{\pi}{4g}\right) & \exp\left(\frac{ik\pi}{2g}\right)\sqrt{\cot^2\left(\frac{\pi}{4g}\right) - 1} \\ \exp\left(-\frac{ik\pi}{2g}\right)\sqrt{\cot^2\left(\frac{\pi}{4g}\right) - 1} & \cot\left(\frac{\pi}{4g}\right) \end{bmatrix}. \tag{II.17}$$

The translation length of each generator is therefore equal to

$$\ell(T) = 2 \cdot \operatorname{arcosh}\left(\cot\left(\frac{\pi}{4g}\right)\right).$$

The systole of a compact surface is generally not known, and is extremely hard to compute. For the family of surfaces $\mathbb{M}_g$, however, we have an explicit formula due to Ebbens and Vegter.

Theorem II.18 ([EV])**.** *The systole of a symmetric hyperbolic surface* $\mathbb{M}_g$ *of genus* g *is given by the formula*

$$\operatorname{sys}(\mathbb{M}_g) = 2 \cdot \operatorname{arcosh}\left(1 + 2\cos\left(\frac{\pi}{2g}\right)\right). \tag{II.19}$$

For closed orientable surfaces, each closed nontrivial geodesic on the surface corresponds to a translation of its fundamental group, and vice versa. To see this, consider the following: let T be a translation of Γ_g, and let p be a point on $\mathrm{Ax}(T)$. T moves p along $\mathrm{Ax}(T)$ (which is a geodesic in $\mathbb{H}^2$), and by definition p and Tp are in the same orbit. The geodesic segment $[p, Tp]$ intersects finitely many copies of D_g, and the intersection of $[p, Tp]$ with each copy of D_g can be translated so that it lies in D_g. Since the end points of this translated segment coincide (as they lie in the same orbit), the projection of $[p, Tp]$ on $\mathbb{M}_g$ is indeed a closed geodesic on $\mathbb{M}_g$. See Figure II.10 for an illustration in genus 2.

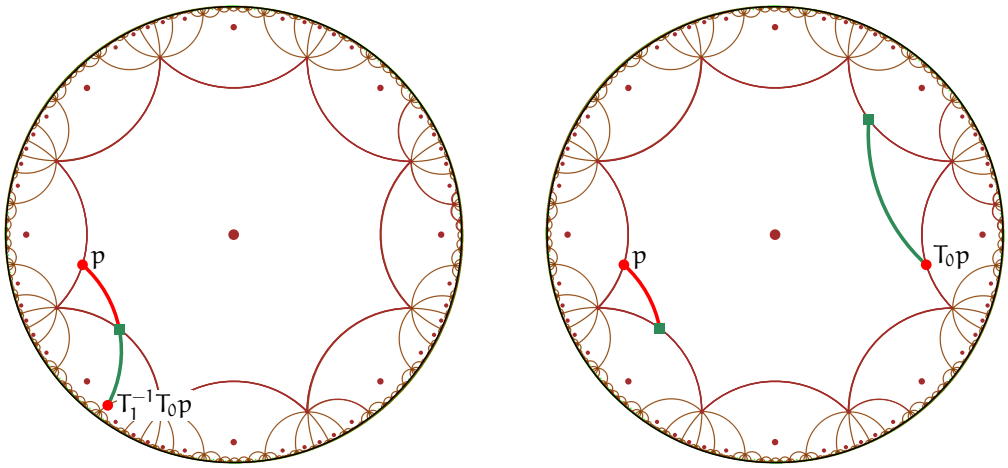

Figure II.10: Geodesic loop on the symmetric hyperbolic surface of genus 2. We have applied the translation $T_1^{-1}T_0$ (arbitrarily chosen) to the point p, and projected the segment $[p, T_1^{-1}T_0p]$ onto D_g.

The Bolza surface

The Bolza surface $\mathbb{M}_2$ is the symmetric hyperbolic surface of genus 2, and has the largest systole of all surfaces of genus 2.[2] It has been studied extensively because it arises naturally in many applications, and also because it is, in a sense, the *simplest* hyperbolic surface with which one can work. Mathematically, it is not simple at all (it is in fact as degenerate as possible), but its symmetry and the fact that it is easily described and constructed make it a preferred model for a periodic hyperbolic space.

Chapters III and IV are dedicated to the formulation of an algorithm for the computation of Delaunay triangulations of $\mathbb{M}_2$ and to its implementation, respectively, so in this section we give a few preliminary details for $\mathbb{M}_2$.

The matrices of the generators of the Bolza surface, ordered counter-clockwise around

[2] Note that the symmetric surfaces of genus $g > 2$ as we have defined them **do not** have maximal systoles for their respective genera.

the origin O, are given in the form

$$T_k = \begin{bmatrix} 1+\sqrt{2} & \exp\left(\frac{ik\pi}{4}\right)\sqrt{2}\sqrt{1+\sqrt{2}} \\ \exp\left(-\frac{ik\pi}{4}\right)\sqrt{2}\sqrt{1+\sqrt{2}} & 1+\sqrt{2} \end{bmatrix}, \qquad k = 0, \ldots, 7, \tag{II.20}$$

where $T_k = T_{k+4}^{-1}$ with indices taken modulo 8. The images of the origin O under the action of the generators of Γ_2, together with the fundamental domain D_2 of $\mathbb{M}_2$, are shown in Figure II.11.

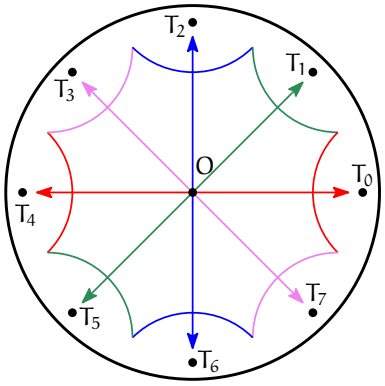

Figure II.11: Generators of the fundamental group of the Bolza surface and their inverses. The figure shows the images of the origin O under the action of the generators of Γ_2.

The fundamental group of Γ_2 has the finite presentation

$$\begin{aligned} \Gamma_2 &= \langle\, T_0, T_1, T_2, T_3 \mid T_0 T_1^{-1} T_2 T_3^{-1} T_0^{-1} T_1 T_2^{-1} T_3 \,\rangle \\ &= \langle\, T_0, T_1, T_2, T_3 \mid T_0 T_5 T_2 T_7 T_4 T_1 T_6 T_3 \,\rangle. \end{aligned} \tag{II.21}$$

The translation length of each generator of Γ_2 is

$$\ell(T_k) = 2 \cdot \operatorname{arcosh}\left(\tfrac{1}{2}\operatorname{Tr}(T_k)\right) = 2 \cdot \operatorname{arcosh}\left(1+\sqrt{2}\right) \approx 3.05714,$$

which is, remarkably, also the value of the systole of $\mathbb{M}_2$, as can be verified from (II.19). Note that this is not true for any other symmetric hyperbolic surface, but, for the Bolza surface in particular,

$$\operatorname{sys}(\mathbb{M}_2) = \ell(T_k).$$

II.2 Triangulations

In this section we introduce Delaunay triangulations, their properties, and ways to compute them. We begin with the definition of triangulation and Delaunay triangulation in the Euclidean plane $\mathbb{E}^2$, which will be the basis for the definition of Delaunay triangulations in $\mathbb{H}^2$ in Section II.3. We then discuss Bowyer's algorithm that enables their construction, and conclude our discussion with Dirichlet regions and their relation to Voronoi diagrams and the fundamental domain of symmetric hyperbolic surfaces.

II.2.1 Delaunay triangulations in the Euclidean plane

Definition II.22 (Triangulation [dBvKOS97]). Let $\mathcal{P}$ be a given finite set of points in the Euclidean plane $\mathbb{E}^2$. A *triangulation* defined by $\mathcal{P}$ is a *maximal planar subdivision* (i.e., a planar subdivision in which no edge can be added without violating its planarity) with vertex set $\mathcal{P}$, or, equivalently, a partition of the convex hull of $\mathcal{P}$ into triangles whose vertices are the points in $\mathcal{P}$.

A triangulation contains both geometric and combinatorial information: it is a combinatorial object whose 1-skeleton is a graph, and it is defined by geometric input (points). In order to distinguish the geometric and the combinatorial elements of a triangulation, we use the following terms. The geometric elements of dimension 0, 1, and 2 of a triangulation are called *point, segment,* and *triangle*, respectively. The corresponding combinatorial elements are called *vertex, edge,* and *face*.

A triangle in a 2-dimensional triangulation whose circumscribing disk is empty (i.e., does not contain any point of the triangulation in its interior) is called a *Delaunay triangle*, or is said to have the *Delaunay property*.

Definition II.23 (Delaunay triangulation). A triangulation of $\mathbb{E}^2$ with vertex set $\mathcal{P}$ is called the *Delaunay triangulation of* $\mathbb{E}^2$ *defined by* $\mathcal{P}$ if each triangle in the triangulation has the Delaunay property.

The Delaunay triangulation defined by a given set of points in $\mathbb{E}^2$ is unique if no more than three points are co-circular, but there are ways to consistently choose one Delaunay triangulation even for sets of points in degenerate position [DT11]. Therefore, throughout the thesis we refer to *the* Delaunay triangulation defined by a set of input points. An important property of a Delaunay triangulation as defined here is that it is a *simplicial complex*.

Definition II.24 (Euclidean simplicial complex [Lee00, Chapter 5]). A Euclidean simplicial complex is a set $\mathcal{K}$ of simplices in some Euclidean space $\mathbb{E}^n$ satisfying the following conditions:

a) If $\sigma \in \mathcal{K}$, then every face of σ is in $\mathcal{K}$.
b) The intersection of any two simplices in $\mathcal{K}$ is either empty or a face of each.
c) Local finiteness: Every point in a simplex of $\mathcal{K}$ has a neighborhood that intersects at most finitely many simplices of $\mathcal{K}$.

Caroli *et al.* [CT16, Proposition 2.1] prove that if the first two conditions of Definition II.24 are satisfied together with the Delaunay property, then local finiteness is guaranteed, i.e., Delaunay triangulations are simplicial complexes. The fact that a Delaunay triangulation is a simplicial complex means that the graph of edges of the triangulation contains no double edges or loops, which in turn means that the set of faces in the triangulation whose circumdisks contain an input point has the topology of a disk.

This property is very important, as it enables us to construct Delaunay triangulations with Bowyer's algorithm, discussed in the next section. An interesting property of the Delaunay triangulation is that it maximizes the minimum angle of its triangles, which makes its triangles less "flat", compared to other triangulations defined by the same set of points. See Figure II.12 for an example of two triangulations in $\mathbb{E}^2$.

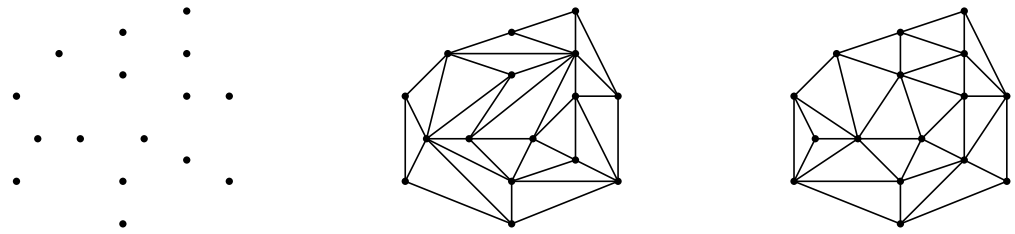

Figure II.12: Comparison of Delaunay (right) and non-Delaunay (center) triangulation defined by the same set of points in the plane (left).

II.2.2 Bowyer's incremental algorithm

There exist various algorithms to compute Delaunay triangulations of $\mathbb{E}^2$. These algorithms are generally separated in three categories: divide-and-conquer [GS85, Dwy87], sweepline [For87], and incremental [Law77, Bow81, Wat81]. The incremental algorithms are arguably the easiest to implement. We will discuss predicates in Section III.4; for the time being, we mention simply that predicates provide robustness at the cost of potentially high-degree algebraic computations, so using predicates of lower degree means that the algorithm is more efficient.

From the category of incremental algorithms, we single out the one proposed by Bowyer [Bow81, Wat81]. Bowyer's algorithm is not only easy to implement, but it also uses only two *predicates* of minimal degree. This particular algorithm has been implemented in CGAL for the computation of Delaunay triangulations of $\mathbb{E}^2$ and $\mathbb{E}^3$ and has been thoroughly tested in many practical applications. Bowyer's incremental algorithm can be summarized in the following steps:

Given a set of points $\mathcal{P} \in \mathbb{E}^2$ and the Delaunay triangulation $\mathcal{T}$ defined by $\mathcal{P}' \subset \mathcal{P}$ such that $|\mathcal{P}'| > 3$:

a) Let $p \in \mathcal{P} \setminus \mathcal{P}'$ be a point to be inserted in $\mathcal{T}$;

b) Find the set of faces Z of $\mathcal{T}$ whose circumdisks contain p;

c) For each face $\sigma \in Z$, destroy σ;

d) Repair the resulting "hole" by forming new faces with p and each edge of the boundary.

An illustration of the algorithm is given in Figure II.13.

Note that the algorithm works for triangulations with more than 3 points in general

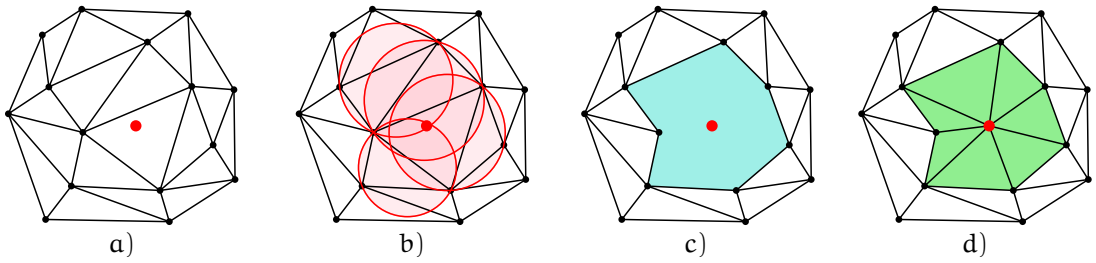

Figure II.13: Insertion of a point in a Delaunay triangulation with Bowyer's incremental algorithm.

position (i.e., not all co-circular); recall, however, that there are methods to deal with points in degenerate position [DT11]. A triangulation defined by 1, 2, or 3 non-collinear points is trivially a single point, a segment, or a triangle, respectively. An important characteristic of Bowyer's algorithm is that it works because the hole created by destroying the faces in Z has the topology of a disk. As we mentioned in Section II.2.1, this property is guaranteed if and only if the triangulation is a simplicial complex.

II.2.3 Dirichlet regions and Voronoi diagrams

We conclude our discussion about triangulations with Voronoi diagrams and Dirichlet regions, which will give us deeper understanding of the fundamental domain of the flat torus $\mathbb{M}_1$. We begin with the definition of Voronoi diagram, which is the *dual* of a Delaunay triangulation.

Definition II.25 (Voronoi cell [dBvKOS97, Chapter 7])**.** Let P be a given set of points in $\mathbb{E}^2$. The *Voronoi cell* V_p of a point $p \in P$ is the set

$$V_p = \left\{ z \in \mathbb{E}^2 \mid d_{\mathbb{E}}(z,p) \leq d_{\mathbb{E}}(z,q), \qquad q \in P \right\}.$$

Definition II.26 (Voronoi diagram [dBvKOS97, Chapter 7])**.** The *Voronoi diagram* $VD_{\mathbb{E}^2}(P)$ of a set of points P in $\mathbb{E}^2$ is the set of *Voronoi cells* $\{ V_p \mid p \in P \}$.

Let p be a point in $\mathbb{E}^2$. The Voronoi diagram of the orbit $\Gamma_1 p$ of p under the action of Γ_1 is a tiling of $\mathbb{E}^2$ similar to the one shown in Figure II.5. The Voronoi cell of p in this Voronoi diagram is called the *Dirichlet region* of p for the group Γ_1.

II.3 Triangulations of closed orientable surfaces

In this section we discuss existing algorithms and implementations for the computation of Delaunay triangulations of closed, compact, orientable surfaces. We begin with triangulations of the flat torus (recall Section II.1.3). Our discussion on this topic is based

on the work of Caroli *et al.* [Car10, CT16]. We continue with generalizations from Euclidean to hyperbolic spaces, which is the work of Bogdanov *et al.* [BTV16]. Finally, we will focus specifically on the Bolza surface and present details for the algorithm proposed by Bogdanov *et al.* in this context.

Before we start our discussion for triangulations of the flat torus, we give the definition of Delaunay triangulation of a compact orientable surface. Initially, Caroli gave a definition for Delaunay triangulation of the flat torus [CT16], and Bogdanov *et al.* extended it to hyperbolic surfaces [BTV16]; in Definition II.27 we combine the two definitions.

Definition II.27 (Delaunay triangulation of compact orientable surface with non-positive Gaussian curvature)**.** Let $\mathbb{X}$ be either $\mathbb{E}^2$ or $\mathbb{H}^2$, and let Γ be a discrete group of orientation-preserving isometries acting on $\mathbb{X}$. Let $\mathbb{M} = \mathbb{X}/\Gamma$ be the compact orientable surface with fundamental group Γ, with natural projection mapping $\pi : \mathbb{X} \to \mathbb{M}$. Let $\mathcal{P}$ be a set of points in $\mathbb{X}$. If the projection under π of the infinite Delaunay triangulation $\mathrm{DT}_{\mathbb{X}}(\Gamma\mathcal{P})$ is a simplicial complex, then it is the Delaunay triangulation of $\mathbb{M}$ defined by $\mathcal{P}$:

$$\mathrm{DT}_{\mathbb{M}}(\mathcal{P}) = \pi(\mathrm{DT}_{\mathbb{X}}(\Gamma\mathcal{P})). \tag{II.28}$$

II.3.1 Triangulations of the flat torus

Caroli works in the Euclidean 3-dimensional space, but in this section we will present his results in two dimensions as the conversion is straightforward. So, for the rest of this section, we work in the Euclidean plane $\mathbb{E}^2$.

Recall the construction of the symmetric flat torus $\mathbb{M}_1$ from a square in $\mathbb{E}^2$, illustrated in Figure II.5. The fundamental group Γ_1 of $\mathbb{M}_1$ is generated by the two axis-aligned unit translations T_x and T_y and has the finite presentation

$$\Gamma_1 = \langle\, T_x, T_y \mid T_x T_y T_x^{-1} T_y^{-1} \,\rangle.$$

$\mathbb{M}_1$ is the quotient space of $\mathbb{E}^2$ under the action of Γ_1, i.e.,

$$\mathbb{M}_1 = \mathbb{E}^2/\Gamma_1,$$

with corresponding natural projection mapping $\pi_1 : \mathbb{E}^2 \to \mathbb{M}_1$. Note that the relation of Γ_1 is trivial because the group is Abelian.

Recall that from Section II.1.6 that the fundamental domain D_1 of a surface contains on its boundary more than one representative of each orbit under the action of Γ_1. In order to have a unique representative for each orbit under Γ_1, Caroli introduces the *original domain* for $\mathbb{M}_1$, which is the half-open square $\mathscr{D}_1 = [0, 1) \times [0, 1)$. Two points $p_1, p_2 \in \mathbb{E}^2$ are *periodic copies* of each other if they lie in the same orbit under the action of Γ_1, or, in other words, if there exists a point $p \in \mathscr{D}_1$ such that $\pi_1(p_1) = \pi_1(p_2) = \pi_1(p)$.

Let $\mathcal{P}$ be a set of points in $\mathscr{D}_1$. Note that this is not a restriction, since $\mathscr{D}_1$ contains exactly one representative of each orbit under Γ_1. Delaunay triangulations of the flat torus are defined as in Definition II.27. However, not all sets of points are such that the projection of $\mathrm{DT}_{\mathbb{E}^2}(\Gamma_1\mathcal{P})$ under π_1 is a simplicial complex. Caroli gives a sufficient geometric condition for an input set of points $\mathcal{P}$ to define a triangulation of $\mathbb{M}_1$. To state this condition, we introduce the following notation. For a set of points $\mathcal{P} \subset \mathscr{D}_1$, let $\Delta(\mathcal{P})$ be the diameter of the largest empty disks in $\Gamma_1\mathcal{P}$, i.e.,

$$\Delta(\mathcal{P}) = \max \text{ diameter of disks not containing any point in } \Gamma_1\mathcal{P}. \tag{II.29}$$

Note that, by definition, $\Delta(\mathcal{P})$ is the circumdiameter of the largest empty disks in $\mathrm{DT}_{\mathbb{E}^2}(\Gamma_1\mathcal{P})$.

Proposition II.30 ([CT16, Criterion 3.11]). *For a given set of points $\mathcal{Q}_1 \subset \mathscr{D}_1$, if*

$$\Delta(\mathcal{Q}_1) < \tfrac{1}{2}\,\mathrm{sys}(\mathbb{M}_1), \tag{II.31}$$

then $\pi_1(\mathrm{DT}_{\mathbb{E}^2}(\Gamma_1\mathcal{S}))$ is a triangulation of $\mathbb{M}_1$ for any finite set $\mathcal{S} \supseteq \mathcal{Q}_1$.

If condition (II.31) is satisfied, then the projection under π_1 of $\mathrm{DT}_{\mathbb{E}^2}(\Gamma_1\mathcal{S})$ is a simplicial complex, i.e., its 1-skeleton does not contain cycles of length one or two. This enables us to use Bowyer's algorithm (recall Section II.2.2). In the actual formulation of Proposition II.30, Caroli uses the least distance by which Γ_1 translates a point (which equals the side length of D_1) instead of the systole of $\mathbb{M}_1$. However, from our discussion about geodesic loops on closed surfaces in Section II.1.6, it is clear that these two quantities refer to the same thing.

A very important implication of Proposition II.30 is that if condition (II.31) is satisfied for a set of points $\mathcal{P}$, adding new points will not violate the condition. This property enables the use of the incremental algorithm that we discussed in Section II.2.2.

If a given set of points $\mathcal{P}$ does not satisfy the validity condition (II.31), there are two solutions: we can either decrease $\Delta(\mathcal{P})$, or we can, in a sense, increase $\mathrm{sys}(\mathbb{M}_1)$. We discuss these two approaches that have been studied by Caroli.

Covering spaces

We begin by giving the general definition of covering space.

Definition II.32 (Covering space [Arm83, Section 10.4]). Let $\mathbb{Y}$ be a topological space. A map $\widetilde{\pi} : \widetilde{\mathbb{Y}} \to \mathbb{Y}$ is called a *covering map* and $\widetilde{\mathbb{Y}}$ is said to be a *covering space* of $\mathbb{Y}$ if the following condition holds. For each point $p \in \mathbb{Y}$ there is an open neighborhood V, and a decomposition of $\widetilde{\pi}^{-1}(V)$ as a family $\{U_\alpha\}$ of pairwise disjoint open subsets of $\widetilde{\mathbb{Y}}$, in such a way that the restriction of $\widetilde{\pi}$ to each U_α is a homeomorphism from U_α to V.

Particularly for compact orientable surfaces with non-positive Gaussian curvature, there is another way to see covering spaces. Let $\mathbb{M} = \mathbb{X}/\Gamma$ with $\mathbb{X}$ either $\mathbb{E}^2$ or $\mathbb{H}^2$ be a compact orientable surface with fundamental group Γ. Let $\widetilde{\Gamma}$ be a normal subgroup of finite index k in Γ. $\widetilde{\mathbb{M}} = \mathbb{X}/\widetilde{\Gamma}$ is a compact orientable surface with natural projection map $\widetilde{\pi} : \mathbb{X} \to \widetilde{\mathbb{M}}$. The surface $\widetilde{\mathbb{M}}$ is a k-sheeted covering space of $\mathbb{M}$ with covering map and local isometry $f : \widetilde{\mathbb{M}} \to \mathbb{M}$ such that $\pi = f \circ \widetilde{\pi}$.

We return now to the flat torus with the following result by Caroli.

Theorem II.33 ([CT16, Theorem 4.2]). *There is a normal subgroup $\widetilde{\Gamma}_1$ of Γ_1 of finite index such that the projection of the Delaunay triangulation of $\Gamma_1\mathcal{P} \bigcup \widetilde{\Gamma}_1 Q$ in $\mathbb{E}^2$ onto $\widetilde{\mathbb{M}}_1 = \mathbb{E}^2/\widetilde{\Gamma}_1$ is a triangulation for any non-empty finite point set $\mathcal{P} \in \mathbb{E}^2$ and any $Q \subseteq \Gamma_1 q$ with any $q \in \mathbb{E}^2$.*

Theorem II.33 implies that there exists a covering space of $\mathbb{M}_1$ with systole large enough so that for any non-empty, finite set of points $\mathcal{P}$ condition (II.31) is always satisfied. Caroli gives a 9-sheeted covering space of $\mathbb{M}_1$ (shown in Figure II.14 - Left) in which condition (II.31) is satisfied for any set of points $\mathcal{P}$ [CT16]. Bogdanov *et al.* later gave another suitable covering space with 8 sheets (shown in Figure II.14 - Right) [BTV16].

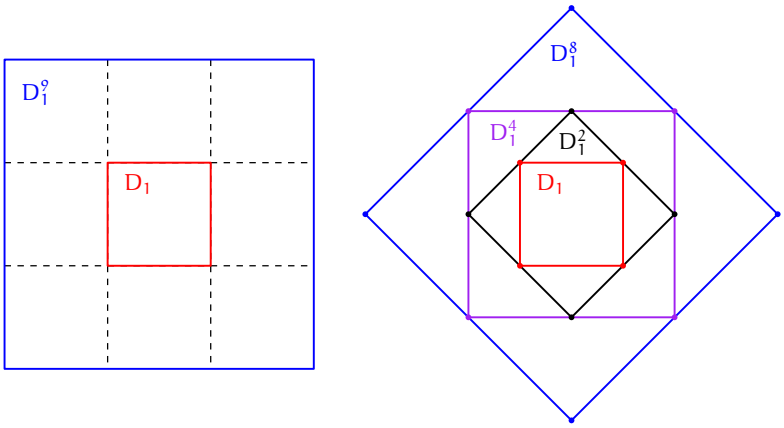

Figure II.14: ***Left:*** *Fundamental domain of the covering space of $\mathbb{M}_1$ with 9 sheets in which any non-empty, finite set of input points in $\mathbb{E}^2$ defines a Delaunay triangulation.* ***Right:*** *Fundamental domain of the covering space of $\mathbb{M}_1$ with 8 sheets with the same property.*

Dummy points

Let $\mathcal{P}$ be a set of points in $\mathscr{D}_1$ that does not satisfy condition (II.31). Instead of using covering spaces to increase the systole, we can instead decrease $\Delta(\mathcal{P})$. The idea is to use a set of *dummy points* $\mathcal{Q}_1$ that satisfies the validity condition (II.31). Then, for any given set of points $\mathcal{P}$, the set of points $\mathcal{S} = \mathcal{P} \bigcup \mathcal{Q}_1$ satisfies the validity condition due to Proposition II.30.

The difference in practice between working on covering spaces or with dummy points is the following.

Working with covering spaces of $\mathbb{M}_1$ requires computations with periodic copies of the input points $\mathcal{P}$ in each sheet of the covering space. So, for the flat torus we need to consider 8 or 9 copies of each input point, if $\mathcal{P}$ does not satisfy condition (II.31).

On the other hand, using dummy points enables us to work directly on $\mathbb{M}_1$ without considering multiple copies of the input points. A triangulation of $\mathbb{M}_1$ is constructed using the set of dummy points, and then the points in $\mathcal{P}$ are inserted one by one with Bowyer's incremental algorithm. On the downside, the existence of "artificial" dummy points in the triangulation might be undesirable or even problematic for some applications.

Caroli details and implements both solutions for triangulations of the 3D flat torus (in which case he works on a covering space with 27 sheets), and an implementation followed for the 2D flat torus on the 9-sheeted covering space shown in Figure II.14 - Left. Both implementations are available in CGAL [CT09, Kru13].

II.3.2 Triangulations of compact orientable hyperbolic surfaces

To define Delaunay triangulations of compact orientable hyperbolic surfaces, we begin by defining Delaunay simplicial complexes in $\mathbb{H}^2$. Recall Definition II.24.

Proposition II.34 ([BDT14, Proposition 4]). *For a set of points $\mathcal{P} \in \mathbb{H}^2$ the Delaunay simplicial complex $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{P})$ defined by $\mathcal{P}$ as points in the Poincaré disk model is a connected simplicial sub-complex of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P})$, defined by the points $\mathcal{P}$ seen as points in the Euclidean unit disk. A face of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P})$ is also a face of $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{P})$ if its circumscribing disk is included in $\mathbb{H}^2$. An edge of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P})$ is also an edge of $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{P})$ if it admits at least one empty disk passing though its vertices and included in $\mathbb{H}^2$.*

For a comparison of a Delaunay simplicial complex in $\mathbb{H}^2$ and its Euclidean counterpart, see Figure II.15.

The tiling of $\mathbb{H}^2$ shown in Figure II.8 is the Voronoi diagram of the orbit $\Gamma_1 O$ of O under the action of Γ_1, and its dual is the Delaunay triangulation $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_1 O)$. The regular hyperbolic polygon D_1 is thus the Voronoi cell of the origin in $\mathrm{VD}_{\mathbb{H}^2}(\Gamma_1 O)$ (equivalently, the Dirichlet region of O for the group Γ_1), and the sides of D_1 are the hyperbolic bisectors of O and its adjacent images in $\Gamma_1 O$. It is important to note that $D_O(\Gamma_1)$ is indeed a fundamental domain for $\mathbb{M}_1$, but it is not *the only* fundamental domain for $\mathbb{M}_1$. We choose to use the Dirichlet region due to its symmetry and its properties, which are essential in the proofs of Theorem III.2 and Proposition V.5, as we will see in the sequel.

Let $\mathbb{M}$ be a closed, orientable, not necessarily symmetric, hyperbolic surface of genus $g \geq 2$. Let D be a fundamental domain of $\mathbb{M}$, and let Γ be its fundamental group. The Delaunay triangulation of $\mathbb{M}$ defined by a set of input points $\mathcal{P}$ is defined as in

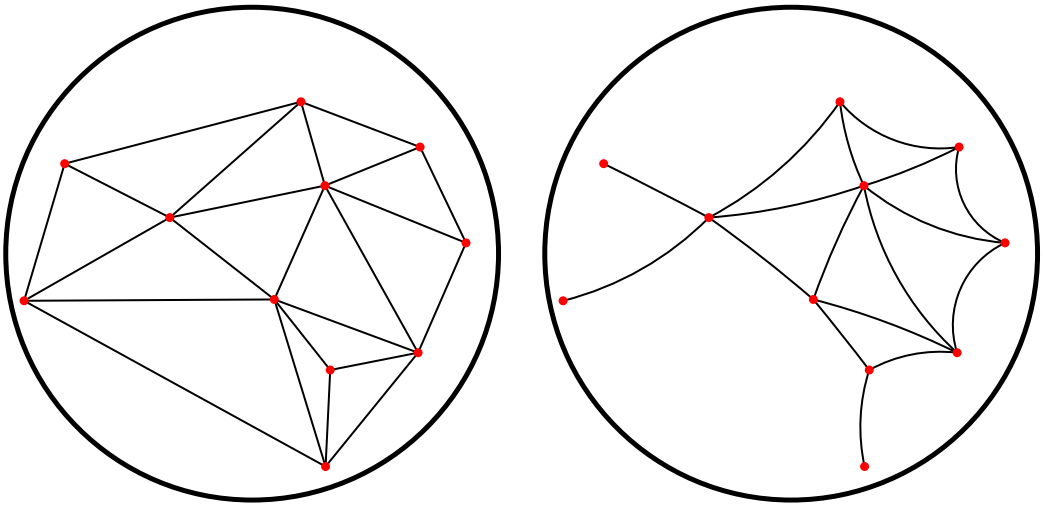

Figure II.15: ***Left:*** *Euclidean Delaunay triangulation of a set of points in the unit disk.* ***Right:*** *Hyperbolic Delaunay complex defined by the same set of points in the Poincaré disk model of the hyperbolic plane.*

Definition II.27. As for the torus, not all sets of input points $\mathcal{P}$ define a triangulation of $\mathbb{M}$, so, inspired by the results of Caroli, Bogdanov *et al.* give a sufficient validity condition similar to Proposition II.30.

Proposition II.35 (Validity condition [BTV16])**.** *Let $\mathcal{P}$ be a set of points in $\mathbb{H}^2$. If*

$$\Delta(\mathcal{P}) < \tfrac{1}{2}\,\mathrm{sys}(\mathbb{M}), \tag{II.36}$$

then the 1-skeleton of $\pi(\mathrm{DT}_{\mathbb{H}^2}(\Gamma\mathcal{S}))$ for any finite set $\mathcal{S}$ containing $\mathcal{P}$ does not contain cycles of length one or two, i.e., the projection is a simplicial complex.

As in the case of the flat torus, in order to satisfy condition (II.36) for any set of points $\mathcal{P} \in \mathbb{H}^2$, we can either increase the systole (by working in a covering space of $\mathbb{M}$), or decrease the diameter of $\mathcal{P}$ (with the use of dummy points). These cases have been studied in particular for the Bolza surface, which we discuss in the next section.

II.3.3 Triangulations of the Bolza surface

Bogdanov *et al.* examine how the validity condition (II.36) can be satisfied [BTV16]. They study the case of the Bolza surface $\mathbb{M}_2$, for which they examine the possibility of working on a covering space, or with a set of dummy points.

Working on a covering space

The study of covering spaces for $\mathbb{M}_2$ revealed that, in order to satisfy condition (II.36) for any set of input points $\mathcal{P}$, one needs to work in a covering space with more than 32

sheets. Ebbens recently proved that in fact the number of necessary sheets is larger than 34 [Ebb17]. Bogdanov *et al.* exhibited a covering space with 128 sheets in which the validity condition is satisfied for any set of input points. It remains currently unknown whether there exists a suitable covering space with fewer than 128 sheets.

The results of Bogdanov *et al.* imply that the idea of using covering spaces is rather impractical. For the Bolza surface, even if we knew that a 35-sheeted covering space is large enough to satisfy condition (II.36) for any set of points, the computational cost would still be very high. For symmetric hyperbolic surfaces of higher genus, Ebbens gives a lower bound of $\frac{256}{3\pi^3} \cdot g^3$ sheets [Ebb17].

Dummy points

The set of dummy points for $\mathbb{M}_2$ is chosen to preserve the symmetries of the surface. The authors propose a set of 14 dummy points $\mathcal{Q}_2$ that satisfy the validity condition (II.36). The set $\mathcal{Q}_2$ consists of the origin O, one vertex of D_2, the midpoints of four consecutive sides of D_2, and the hyperbolic midpoints of the segments from O to the eight vertices of D_2. The dummy points for the Bolza surface, as well as the Delaunay triangulation of $\mathbb{M}_2$ that they define, are shown in Figure II.16.

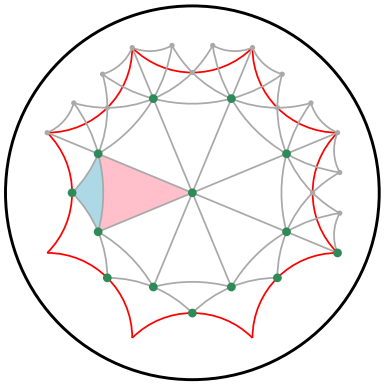

Figure II.16: Delaunay triangulation of $\mathbb{M}_2$ defined by the dummy points $\mathcal{Q}_2$. The triangulation consists of only two types of triangles (filled in the figure). All other triangles can be obtained from these two with reflections and rotations.

In this thesis we focus our efforts on computing Delaunay triangulations of symmetric hyperbolic surfaces with the use of dummy points.

Chapter III

Delaunay triangulations of the Bolza surface

In Section II.3.2 we saw an algorithm proposed by Bogdanov, Teillaud and Vegter for the computation of triangulations of the Bolza surface $\mathbb{M}_2$ with the use of dummy points [BTV16]. In this chapter we discuss in detail a data structure and operations on it that we propose in order to implement this algorithm. We begin by discussing the representation of the triangulation. We will then discuss how the triangulation of the set of proposed dummy points is constructed, and how input points are subsequently inserted. We also discuss the location of a point in the triangulation and the removal of existing vertices. In the last section of this chapter, we discuss the predicates that we use in our algorithm and their algebraic complexity.

The larger part of the material in this chapter has already been published [IT17].[3] However, material that could not be included in the paper due to lack of space has been integrated in this chapter. These additions are annotated properly throughout the chapter. The implementation in CGAL of the algorithm that we discuss here is presented in Chapter IV.

III.1 Representation of the triangulation

Recall from Section II.3.2 the definition of the closed regular octagon D_2 in $\mathbb{H}^2$ that is a fundamental domain for $\mathbb{M}_2$. D_2 contains at least one preimage of each point on $\mathbb{M}_2$, with duplicate representatives on the boundary of D_2. We introduce an *original domain* $\mathscr{D}_2 \subset D_2$ for $\mathbb{M}_2$ that contains exactly one point of the fiber under the projection map π_2 of each point on the surface $\mathbb{M}_2$: (see Figure III.1) $\mathscr{D}_2$ consists of the interior of D_2, its four "solid" sides, and one vertex of the octagon (chosen to be V_0). Note that $\pi_2(V_k) = \pi_2(V_0)$, $k = 1, \ldots, 7$: $V_5 = T_0^{-1}V_0, V_2 = T_1V_5, V_7 = T_2^{-1}V_2, V_4 = T_3^{-1}V_2, V_1 =$

[3] Note that the generators of Γ_2 are denoted differently here than in [IT17]; the correspondence is the following: $a = T_0, b^{-1} = T_1, c = T_2, d^{-1} = T_3, a^{-1} = T_4, b = T_5, c^{-1} = T_6, d = T_7$.

$T_0V_4, V_6 = T_1^{-1}V_1, V_3 = T_2V_6, V_0 = T_3^{-1}V_3$.

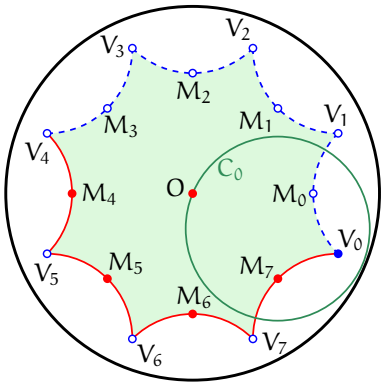

Figure III.1: Enumeration of the vertices of D_2 *and the midpoints of its sides. The original domain* $\mathscr{D}_2$ *contains the interior of* D_2*, the four sides* $[V_i, V_{i+1}], i = 4, 5, 6, 7$*, and the vertex* V_0.

Recall from Section II.3.2 that the use of dummy points allows us to always assume that **the set** $\mathcal{S} = \mathcal{P} \bigcup \mathcal{Q}_2$ **satisfies inequality** (II.36) **for any set of input points** $\mathcal{P}$. Note that all dummy points lie in $\mathscr{D}_2$, and we can consider without loss of generality that all points of $\mathcal{P}$ lie in $\mathscr{D}_2$; by consequence, from now on we will consider that all points in $\mathcal{S}$ lie in $\mathscr{D}_2$. Similarly, we will now define a unique representative in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$ for each face of the Delaunay triangulation $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ of $\mathbb{M}_2$ defined by $\mathcal{S}$.

III.1.1 Canonical representative of a face

The definition of the canonical representative of a face will rely on Theorem III.2, which is reminiscent of the result proved for the flat torus by Dolbilin and Huson [DH97] and recalled in [CT16, Lemma 6.3].

We denote the hyperbolic distance between two points p and q in $\mathbb{H}^2$ as $d_{\mathbb{H}}(p, q)$ and the (hyperbolic) segment with endpoints p and q as $[p, q]$. We introduce an abuse of notation: T denotes both a translation of Γ_2 and the point TO (i.e., the image of the origin O under the translation T); recall Figure II.11. The points M_k, $k = 0, \ldots, 7$ visible on Figure III.1 are defined as the midpoints of V_k and V_{k+1} (indices are meant modulo 8).

We denote with C_k the hyperbolic circle that passes through O and is centered at the vertex V_k of D_2. Let $\mathcal{U}_{D_2}$ be the union of the disks bounded by the circles $C_k, k = 0, 1, \ldots, 7$, and let $\mathcal{C}_{D_2}$ be the boundary of $\mathcal{U}_{D_2}$. See figure III.2-Left.

Lemma III.1. *The distance between* $\mathcal{C}_{D_2}$ *and* ∂D_2 *is equal to* $d_{\mathbb{H}}(M_k, T_k) = \frac{1}{2}\,\mathrm{sys}(\mathbb{M}_2)$, $k = 0, 1, \ldots, 7$.

Proof. Using symmetries, we get (see figure III.2-Right):

$$d_{\mathbb{H}}(\mathcal{C}_{D_2}, \partial D_2) = \min_{p \in [M_k, V_k],\ q \in \mathcal{C}_{D_2} \cap C_k} d_{\mathbb{H}}(p, q).$$

For any point $q \in \mathcal{C}_{D_2} \cap C_k$, let p be the projection of q onto $[M_k, V_k]$, so that the length of the segment $[p, q]$ is the distance between q and $[M_k, V_k]$, as shown in Figure III.2 - Right. For $q = T_k$, the projection of q onto $[M_k, V_k]$ is the point $p = M_k$. Moreover, for all $q \in \mathcal{C}_{D_2} \cap C_k$, the length of the segment $[q, V_k]$ is constant, as V_k is the hyperbolic center of C_k. In particular, let q_1 and q_2 be the two points on C_k (so that O, q_1, and q_2 are oriented counter-clockwise) such that the segments $[q_i, V_k], i = 1, 2$ are perpendicular to both $[M_k, V_k]$ and C_k (see Figure III.2 - Right); for $q \in C_k$ between q_1 and q_2, V_k projects onto q since $[q, V_k]$ is perpendicular to C_k, so the distance between $[M_k, V_k]$ and q equals the radius of C_k.

Consider now that q starts from T_k and moves continuously on the curve $\mathcal{C}_{D_2} \cap C_k$ until it reaches q_1. The point p then moves continuously from M_k to V_k, and the angle (p, V_k, q) also grows continuously until p reaches V_k and becomes a right angle. The length of the segment $[p, q]$ also grows continuously: the hypotenuse $[V_k, q]$ of the right triangle (V_k, q, p) has constant length since q lies on C_k centered at V_k, and the angle (q, p, V_k) is right for all q; the Hyperbolic law of sines thus implies that

$$\frac{\sin(\text{angle}(q, p, V_k))}{\sinh(d_{\mathbb{H}}(V_k, q))} = \text{constant} = \frac{\sin(\text{angle}(p, V_k, q))}{\sinh(d_{\mathbb{H}}(p, q))}$$

The result follows from the fact that the minimal angle (p, V_k, q) is obtained for $q = T_k$ and $p = M_k$.

□

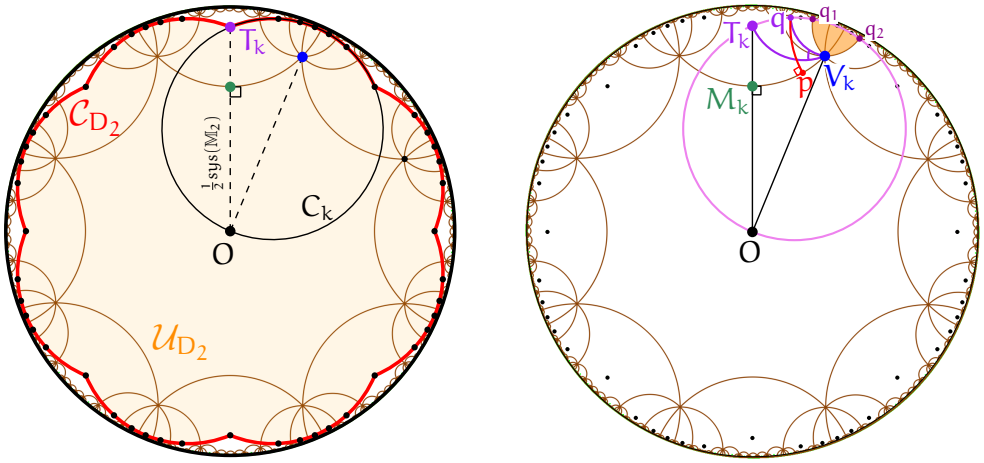

Figure III.2: ***Left:*** *Curve $\mathcal{C}_{D_2}$ (in bold).* ***Right:*** *The distance between $\mathcal{C}_{D_2}$ and ∂D_2 is realized as the distance between the points T_k and M_k.*

For $T \in \Gamma_2$, let TD_2 denote the closure of the region of T in $VD_{\mathbb{H}}(\Gamma_2 O)$; TD_2 is the image of D_2 by the translation T. The infinite set of regions TD_2, for $T \in \Gamma_2$, form a tiling of the plane $\mathbb{H}^2$ (it was shown on Figure II.8 - Left). We define $\mathcal{N}_2$ as the set of translations T in Γ_2 for which $TD_2 \cap D_2 \neq \emptyset$. The set $\mathcal{N}_2$ has 49 elements; it is naturally ordered counterclockwise around O, following the boundary of D_2. Each element T of $\mathcal{N}_2 \setminus \mathrm{Id}$ has an index $\mathrm{idx}_{\mathcal{N}_2}(T)$ in this sequence, while no index is defined for the identity translation. We choose $T^{\text{first}} = T_0 T_1^{-1} T_2 T_3^{-1}$ as the first element for the sequence $\mathcal{N}_2$, i.e., $\mathrm{idx}_{\mathcal{N}_2}(T^{\text{first}}) = 0$. See figure III.3-Left. We define $D_{\mathcal{N}_2}$ as

$$D_{\mathcal{N}_2} = \bigcup_{T \in \mathcal{N}_2} TD_2.$$

See Figure III.3-Right.

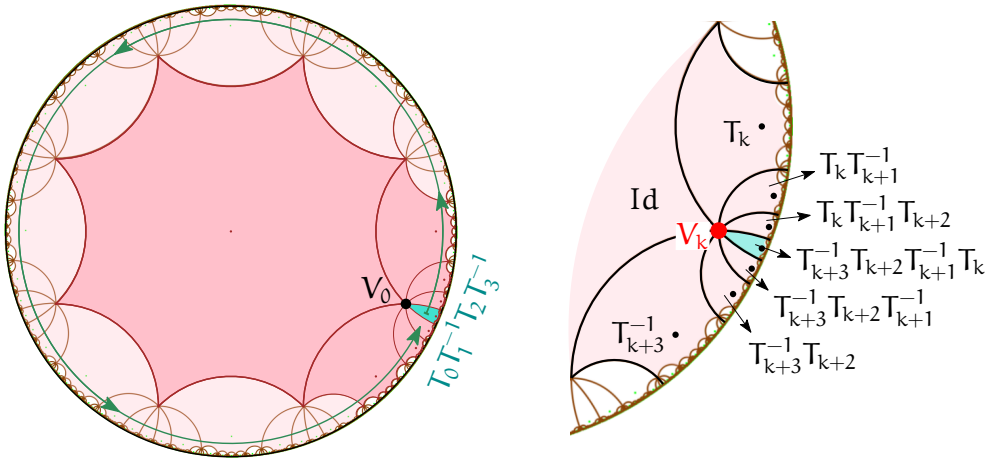

Figure III.3: ***Left:*** *The set $D_{\mathcal{N}_2}$ is the union of the 49 Dirichlet regions that have at least one vertex in common with D_2, shaded in the figure. The ordering of these regions is indicated by the arrow starting from the region $T_0T_1^{-1}T_2T_3^{-1}$.* ***Right:*** *Zoom-in to the vertex V_0, showing the Dirichlet regions incident to it. For any vertex V_k of D_2, the indexing of its incident Dirichlet region is shown in the figure. Note that $T_kT_{k+1}^{-1}T_{k+2}T_{k+3}^{-1} = T_{k+3}^{-1}T_{k+2}T_{k+1}^{-1}T_k$ for indices $k = 0, \ldots, 7$ taken modulo 8.*

Theorem III.2. *Let $\mathcal{S} \subset \mathbb{H}^2$ be a set of points such that inequality (II.36) holds for $\mathbb{M}_2$. If a face σ of $DT_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$ has at least one of its vertices in $\mathscr{D}_2$, then σ is contained in $D_{\mathcal{N}_2}$.*

Proof. Let σ be a face in $DT_{\mathbb{H}^2}(\mathcal{S})$ with at least one vertex in $\mathscr{D}_2$. Recall the definition of $\Delta(\mathcal{S})$ (II.29). The circumscribing disk of σ has diameter smaller than $\Delta(\mathcal{S})$, which is smaller than $\frac{1}{2}\mathrm{sys}(\mathbb{M}_2)$ by inequality (II.36). Lemma III.1 allows us to conclude that this disk is contained in $\mathcal{U}_{D_2}$.

We will now prove that $\mathcal{U}_{D_2}$ is contained in $D_{\mathcal{N}_2}$, by proving that each circle C_k, for $k \in \{0, 1, \ldots, 7\}$ is contained in $D_{\mathcal{N}_2}$. A circle C_k is centered at the Voronoi vertex

V_k; it passes through the origin O and its images under the action of seven consecutive elements of $\mathcal{N}_2$. Rotating C_k around V_k by $\frac{\pi}{4}$ maps each of these eight points (and its Voronoi region) to the next one along C_k. This rotational symmetry shows that in order to prove that $C_k \subset D_{\mathcal{N}_2}$, it is enough to prove that C_k intersects only the two sides of D_2 that are incident to its hyperbolic center V_k.

Indices below are again taken modulo eight, e.g., we write V_{k+1} instead of $V_{k+1 \bmod 8}$. Let us first show that C_k intersects the sides $[V_{k-1}, V_k]$ and $[V_k, V_{k+1}]$ of D_2. Consider a hyperbolic triangle $\triangle(O, V_k, V_{k+1})$. Its angle at O is $\frac{\pi}{4}$, while the angles at the vertices V_k and V_{k+1} are $\frac{\pi}{8}$. From the Hyperbolic law of sines,[4] we conclude that the length of $[V_k, V_{k+1}]$ is larger than the length of $[O, V_k]$. The result follows, since the segment $[O, V_k]$ is a radius of C_k.

Consider now the line segment $l_k = [V_{k-2}, V_{k+2}], k = 0, 1, \ldots, 7$, which cuts the octagon into two halves. See Figure III.4-Left. Both l_k and C_k contain O; moreover l_k is perpendicular to the segment $[O, V_k]$, which is supported by a diameter of C_k. So l_k and C_k are tangent at O and l_k separates C_k from the other half of the octagon, thus C_k cannot intersect any side $[V_{k+j}, V_{k+j+1}]$ of D_2 for $j = 2, 3, 4, 5$.

We will show now that C_k does not intersect either $[V_{k-2}, V_{k-1}]$ or $[V_{k+1}, V_{k+2}]$. By symmetry, it suffices to consider $[V_{k+1}, V_{k+2}]$. We will show that the distance between V_k and $[V_{k+1}, V_{k+2}]$ exceeds $\frac{1}{2}\operatorname{sys}(\mathbb{M}_2)$.[5]

Any two sides of D_2 not incident to the same vertex are supported by *ultraparallel lines* (lines that are parallel and have no common ideal point). Concretely, the lines supporting $[V_{k-1}, V_k]$ and $[V_{k+1}, V_{k+2}]$ (which we will denote with $\mathcal{L}_{[V_{k-1}, V_k]}$ and $\mathcal{L}_{[V_{k+1}, V_{k+2}]}$, respectively) are ultraparallel; so, there exists a point R_{k-1} on $\mathcal{L}_{[V_{k-1}, V_k]}$ and a point L_{k+1} on $\mathcal{L}_{[V_{k+1}, V_{k+2}]}$ such that the segment $[R_{k-1}, L_{k+1}]$ is perpendicular to both $\mathcal{L}_{[V_{k-1}, V_k]}$ and $\mathcal{L}_{[V_{k+1}, V_{k+2}]}$. See figure III.4 - Right for an illustration. In fact, the point R_{k-1} lies on the segment $[M_{k-1}, V_k]$, and the point L_{k+1} lies on the segment $[V_{k+1}, M_{k+1}]$ due to the following arguments. The segment $[V_k, V_{k+1}]$ meets both lines $\mathcal{L}_{[V_{k+1}, V_{k+2}]}$ and $\mathcal{L}_{[V_{k-1}, V_k]}$ at an angle $\frac{\pi}{4}$. By moving V_k to M_{k-1} and V_{k+1} to M_{k+1}, we obtain the segment $[M_{k-1}, M_{k+1}]$. Since the segment $[M_{k-1}, M_{k+1}]$ splits the right angles (O, M_{k-1}, V_k) and (V_{k+1}, M_{k+1}, O), both angles $(V_{k-1}, M_{k-1}, M_{k+1})$ and $(M_{k-1}, M_{k+1}, V_{k+2})$ are larger than $\frac{\pi}{2}$. By consequence, the line $\mathcal{L}_{[R_{k-1}, L_{k+1}]}$ is perpendicular to both $\mathcal{L}_{[V_{k-1}, V_k]}$ and $\mathcal{L}_{[V_{k+1}, V_{k+1}]}$ for some point $R_{k-1} \in [M_{k-1}, V_k]$ and $L_{k+1} \in [V_{k+1}, M_{k+1}]$.

The polygon $OM_{k-1}R_{k-1}L_{k+1}M_{k+1}$ is a right-angled pentagon. From hyperbolic trigonometry (see, for example, [Bea83, Chapter 7.18]), the length of the side $[R_{k-1}, L_{k+1}]$

[4] From [Bea83, Chapter 7.12]: if A, B, C are the sides of a hyperbolic triangle and $\vartheta_A, \vartheta_B, \vartheta_C$ the angles opposite to each side, then

$$\frac{\sin(\vartheta_A)}{\sinh(A)} = \frac{\sin(\vartheta_B)}{\sinh(B)} = \frac{\sin(\vartheta_C)}{\sinh(C)}.$$

[5] Note that from this point on, the proof is not the same as in [IT17]. The proof in the paper uses Euclidean circles, while here it stays in the hyperbolic plane.

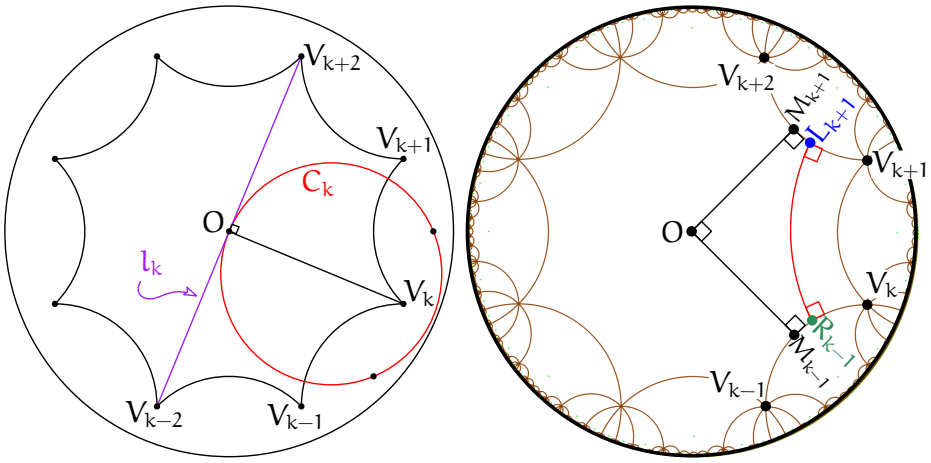

Figure III.4: ***Left:*** C_k *intersects the sides* $[V_{k-1}, V_k]$ *and* $[V_k, V_{k+1}]$*, but cannot intersect any side* $[V_{k+j}, V_{k+j+1}]$ *for* $j = 2, 3, 4, 5$. ***Right:*** *The segment* $[R_{k-1}, L_{k+1}]$ *is perpendicular to both ultraparallel sides* $[V_{k-1}, V_k]$ *and* $[V_{k+1}, V_{k+2}]$ *of* D_2.

can be computed as

$$\cosh(d_{\mathbb{H}}(R_{k-1}, L_{k+1})) = \sinh(d_{\mathbb{H}}(O, M_{k+1}))\sinh(d_{\mathbb{H}}(O, M_{k-1})) = \sinh^2\left(\tfrac{1}{2}\operatorname{sys}(\mathbb{M}_2)\right).$$

By substituting $\frac{1}{2}\operatorname{sys}(\mathbb{M}_2) = \operatorname{arcosh}(1+\sqrt{2})$, and by using the fact that $\sinh^2(\operatorname{arcosh}(x)) = x^2 - 1$, we find that

$$\begin{aligned}\cosh(d_{\mathbb{H}}(R_{k-1}, L_{k+1})) &= \sinh^2\left(\operatorname{arcosh}\left(1+\sqrt{2}\right)\right)\\ &= (1+\sqrt{2})^2 - 1\\ &= 3 + 2\sqrt{2} - 1\\ &= 2(1+\sqrt{2}).\end{aligned}$$

Since $d_{\mathbb{H}}(R_{k-1}, L_{k+1}) = \operatorname{arcosh}(2(1+\sqrt{2}))$ and $\frac{1}{2}\operatorname{sys}(\mathbb{M}_2) = \operatorname{arcosh}\left(1+\sqrt{2}\right)$, we conclude that

$$d_{\mathbb{H}}(R_{k-1}, L_{k+1}) > \tfrac{1}{2}\operatorname{sys}(\mathbb{M}_2).$$

We have shown therefore that

$$d_{\mathbb{H}}\left(V_k, \mathcal{L}_{[V_{k+1}, V_{k+2}]}\right) \geq \min d_{\mathbb{H}}\left(\mathcal{L}_{[V_{k-1}, V_k]}, \mathcal{L}_{[V_{k+1}, V_{k+2}]}\right) = d_{\mathbb{H}}(R_{k-1}, L_{k+1}) > \tfrac{1}{2}\operatorname{sys}(\mathbb{M}_2),$$

which concludes the proof.

□

Let $\mathcal{S} \subset \mathscr{D}_2$ be a set of points satisfying inequality (II.36) for $\mathbb{M}_2$. The rest of this section is dedicated to the choice of a unique *canonical representative* σ^c in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$ for each face $\boldsymbol{\sigma}$ in $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$.

Let $\boldsymbol{\sigma}$ be a face in $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$. By definition of $\mathscr{D}_2$, each vertex of $\boldsymbol{\sigma}$ has a unique preimage by π_2 in $\mathscr{D}_2$, so, the set

$$\Sigma = \left\{ \sigma \in \pi_2^{-1}(\boldsymbol{\sigma}) \;\middle|\; \sigma \text{ has at least one vertex in } \mathscr{D}_2 \right\} \tag{III.3}$$

contains at most three faces. See Figure III.5. When Σ contains only one face, then this face is completely included in $\mathscr{D}_2$, and we naturally choose it to be σ^c. Let us now assume that Σ contains two or three faces. From Theorem III.2, each face $\sigma \in \Sigma$ is contained in $\mathrm{D}_{\mathcal{N}_2}$. So, for each vertex v of σ, there is a unique translation $\mathrm{T}(v,\sigma)$ in $\mathcal{N}_2$ such that v lies in $\mathrm{T}(v,\sigma)\mathscr{D}_2$.

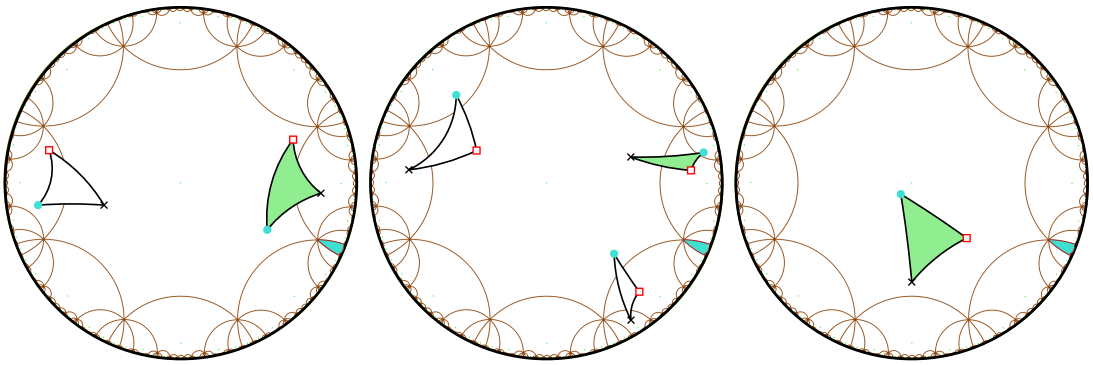

Figure III.5: Examples of faces of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$ with one, two and three vertices in $\mathscr{D}_2$, that project to the same face on $\mathbb{M}_2$. Their respective vertices drawn as a dot project to the same vertex on $\mathbb{M}_2$ (same for cross and square). The canonical representative is the shaded face.

We consider all faces in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$ oriented counterclockwise. For $\sigma \in \Sigma$, we denote as v_σ^{out} the first vertex of σ (in the counterclockwise order) that is not lying in $\mathscr{D}_2$. Using the indexing on $\mathcal{N}_2$ defined above, we can now choose σ^c as the face of Σ whose first vertex lying outside $\mathscr{D}_2$ is "closest" to the region $\mathrm{T}^{\mathrm{first}}\mathrm{D}_2$ in the counterclockwise order around O:

Definition III.4 (Canonical representative)**.** With the notation defined above, the canonical representative of a face $\boldsymbol{\sigma}$ of $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ is the face $\sigma^c \in \Sigma$ such that

$$\mathrm{idx}_{\mathcal{N}_2}\left(\mathrm{T}(v_{\sigma^c}^{\mathrm{out}}, \sigma^c)\right) = \min_{\sigma\in\Sigma} \mathrm{idx}_{\mathcal{N}_2}\left(\mathrm{T}(v_\sigma^{\mathrm{out}}, \sigma)\right).$$

III.1.2 Data structure

For a given set of points $\mathcal{S}$, the data structure that we use stores the faces and vertices of $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$. Each vertex v of $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ represents an orbit under the action of Γ_2; it stores the point of $\pi_2^{-1}(v)$ that belongs to $\mathscr{D}_2$. Recall from Section III.1 that we can

assume that all input points of $\mathcal{S}$ lie in $\mathscr{D}_2$. Moreover, each vertex gives access to one of its incident faces.

Faces $\boldsymbol{\sigma}$ of $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ are stored through their canonical representative σ^c in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$. Each face gives access to its neighboring faces and to its incident vertices. Moreover, each face $\boldsymbol{\sigma}$ stores the three translations $\mathsf{T}(v_i, \sigma^c) \in \mathcal{N}_2$, $i = 0, 1, 2$ defined at the end of Section III.1.1. In this way, for a given face $\boldsymbol{\sigma}$ in the structure, the corresponding canonical representative is the triangle in $\mathbb{H}^2$ whose vertices are the images by $\mathsf{T}(v_i, \sigma^c)$ of the point in $\mathscr{D}_2$ stored in v_i for $i = 0, 1, 2$. The translations $\mathsf{T}(v_i, \sigma^c)$ play a similar role as the so-called "offsets" of the Euclidean periodic triangulations mentioned in Section II.3.1.

Hyperbolic translations have a double nature: they can be seen both as 2×2 matrices with complex coefficients, as we have already seen in Section II.1.1 (recall (II.14)), and as words on the alphabet defined by the generators of Γ_2. The matrix form will be discussed in Section IV.3, while the matrices themselves are given in Table III.16; here we focus on their word representation.

Translations as words

We consider the cyclical sequence $\mathcal{A} = [\mathsf{T}_0, \ldots, \mathsf{T}_7]$ formed by generators of Γ_2 and their inverses as an alphabet, and we denote the set of words on $\mathcal{A}$ as $\mathcal{A}^*$. Each translation T in Γ_2 can be seen as a word in $\mathcal{A}^*$, also denoted as T. For two translations $\mathsf{T}, \mathsf{T}' \in \Gamma_2$, the composition (or multiplication) $\mathsf{T}\mathsf{T}'$ corresponds to the concatenation of the two words T and T'. Recall that composition is not commutative. We have seen in the two previous sections that we only need to store translations in $\mathcal{N}_2$. Let us note here that $\mathcal{N}_2$ is closed under inversion, but not under composition.

The finite presentation of Γ_2 captures the fact that a translation $\mathsf{T} \in \Gamma_2$ does not have a unique representation in terms of the generators (see Section II.3.2). To obtain a unique representation of the translations that are involved in our algorithm, we slightly modify Dehn's algorithm. Dehn's algorithm solves the *word problem* (i.e., the problem of deciding whether a given word on the generators of a group is equal to the group identity) in the case of fundamental groups of closed orientable surfaces of genus at least 2 [Deh12, Gre60].[6]

Let w be a non-trivial word in $\mathcal{A}^*$. Let also $\mathcal{R}_2 = \mathsf{T}_0\mathsf{T}_1^{-1}\mathsf{T}_2\mathsf{T}_3^{-1}\mathsf{T}_0^{-1}\mathsf{T}_1\mathsf{T}_2^{-1}\mathsf{T}_3$ denote the relation of Γ_2. Note that $\mathcal{R}_2^{-1}$, as well as all cyclic permutations of $\mathcal{R}_2$ and $\mathcal{R}_2^{-1}$ are also relations of Γ_2, as they are all equivalent to the identity of Γ_2. This can be viewed in another way by considering $\mathcal{R}_2^\infty$, the infinite word formed by infinitely many concatenations of $\mathcal{R}_2$: any subsequence R of $\mathcal{R}_2^\infty$ or $(\mathcal{R}_2^\infty)^{-1}$ with $\ell_{\mathcal{A}}(\mathsf{R}) = \ell_{\mathcal{A}}(\mathcal{R}_2)$ is a relation in Γ_2, i.e., it reduces to Id. (Here $\ell_{\mathcal{A}}(\cdot)$ denotes the length of a word.)

The first step of Dehn's algorithm consists in *freely reducing* w, i.e., removing all sub-words of the form $\mathsf{T}\mathsf{T}^{-1}$ or $\mathsf{T}^{-1}\mathsf{T}$ for $\mathsf{T} \in \mathcal{A}$. Then, the algorithm executes a second

[6] For interesting historical facts on this topic, see [OR03]. Software solving the word problem (when it can be solved) can be found for instance in [GAP16, mag].

reduction step based on the idea that if the freely reduced w contains a subsequence of $\mathcal{R}_2^\infty$ or $(\mathcal{R}_2^\infty)^{-1}$, then this subsequence can either be reduced to the identity or to a shorter word. This second step is performed by detecting a factorization of the freely-reduced word w of the form $w = w_\lambda w_\mu w_\kappa$, where $w_\mu t$ is a relation R for some $t \in \mathcal{A}^*$ with $\ell_\mathcal{A}(t) < \ell_\mathcal{A}(w_\mu)$. Then $\ell_\mathcal{A}(w_\mu) > \ell_\mathcal{A}(\mathcal{R}_2)/2 = 4$ and w_μ can be substituted in w by t^{-1}, which yields the word $w_\lambda t^{-1} w_\kappa$ with length shorter than $\ell_\mathcal{A}(w)$.

The two steps (free reduction and relation simplification) are repeated until $w = \mathrm{Id}$ or until w cannot be further reduced. In the original algorithm by Dehn, words of length $\ell_\mathcal{A}(\mathcal{R}_2)/2$ are not reduced. In order to have a unique representations of words of length four, we introduce a small modification to the algorithm: whenever we get an irreducible word w with $\ell_\mathcal{A}(w) = 4$, we check whether w is a sub-word of $(\mathcal{R}_2^\infty)^{-1}$. If so, we return w^{-1}; in all other cases, we return w.

Dehn's algorithm terminates in a finite number of steps and its time complexity is polynomial in the length of the input word. Note that we reduce words that are formed by the concatenation of two or three words in $\mathcal{N}_2$; this will become clear in Section III.3.1. Since the longest word in $\mathcal{N}_2$ has four letters, the longest words that we reduce have length 12.

III.2 Construction of the triangulation

In this section we present the operations to build a triangulation of the Bolza surface defined by a given set of points $\mathcal{P}$. We will see how the triangulation is initialized with the dummy points $\mathcal{Q}_2$, how the faces *in conflict* with a new point are identified, and finally how this new point is inserted into the triangulation.

III.2.1 Initialization

The set $\mathcal{Q}_2$ of 14 dummy points proposed in [BTV16, Section 4.2] is as follows:

- the origin O;
- the eight midpoints P_k of the hyperbolic segments $[O, V_k]$, $k = 0, 1, \ldots 7$;
- the midpoints M_k, $k = 4, 5, 6, 7$ of the closed sides of $\mathscr{D}_2$;
- the vertex V_0 of $\mathscr{D}_2$.

The canonical representatives of the 32 faces forming the Delaunay triangulation of $\mathcal{Q}_2$ are shown in Figure III.6-Left. They can be constructed in four iterations ($i = 0, 1, 2, 3$) by using the numbering shown in figure III.6-Right (but faces are not numbered in the code).

The coordinates of the dummy points are algebraic numbers, as reported in Table III.7. They have been computed using Maple.[7] Due to lack of space in [IT17], the calculations were not shown.

[7] Worksheets available at https://members.loria.fr/Monique.Teillaud/DT_Bolza_SoCG17/.

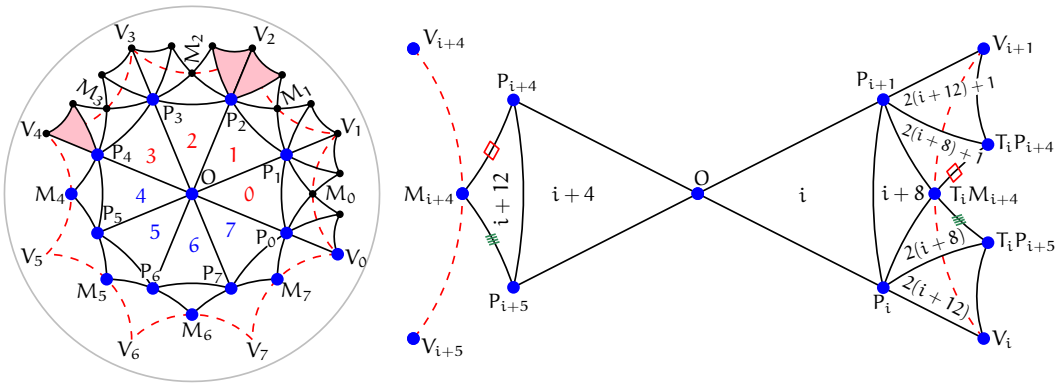

Figure III.6: ***Left:*** *Delaunay triangulation of $\mathbb{M}_2$ defined by the dummy points.* ***Right:*** *Zooming in on the faces created in iteration* i*. Note the identification of the marked edges.*

To compute the coordinates of the vertices and midpoints of D_2, we use the material by Balazs and Voros [BV86, Fig. 16], according to which the Euclidean distance between the origin and the vertex V_0 is $2^{-1/4}$. The vertices of D_2 lie on a circle centered at the origin, the quantity $2^{-1/4}$ is in fact the radius of this circle. Since this circle intersects the real axis at point $\left(2^{-1/4}, 0\right)$, in order to find V_0 it suffices to rotate this point by angle $-\frac{\pi}{8}$. Computing this rotation gives the result

$$V_0 = \left(\frac{2^{3/4}\sqrt{2+\sqrt{2}}}{4}, -\frac{2^{3/4}\sqrt{2-\sqrt{2}}}{4}\right).$$

The other vertices V_k of D_2 can be obtained by rotating V_0 by $\frac{k\pi}{4}$ for $k = 1, \ldots, 7$.

Returning to Balazs and Voros [BV86, Fig. 16], the Euclidean distance between the origin and M_0 is $\sqrt{\sqrt{2}-1}$, therefore

$$M_0 = \left(\sqrt{\sqrt{2}-1}, 0\right).$$

Again, the midpoints M_k can be obtained by rotating M_0 by $\frac{k\pi}{4}$ for $k = 1, \ldots, 7$.

To compute the coordinates of the points P_k, we rely on the fact that the points lie on a circle centered at O. In order to find the Euclidean radius of this circle, we will need to reason first with hyperbolic distances. As we mentioned, P_k is the hyperbolic midpoint of the segment $[O, V_k]$; in other words,

$$d_{\mathbb{H}}(O, P_k) = d_{\mathbb{H}}(P_k, V_k) = \frac{1}{2} d_{\mathbb{H}}(O, V_k). \tag{III.5}$$

The hyperbolic distance from the origin to the point P_k is given by the formula

$$d_{\mathbb{H}}(O, P_k) = \operatorname{arcosh}\left(1 + \frac{2 d_{\mathbb{E}}(O, P_k)^2}{1 - d_{\mathbb{E}}(O, P_k)^2}\right),$$

where $d_{\mathbb{E}}(O, P_k)$ denotes the Euclidean distance between the origin and the point P_k. Thus, from Equation (III.5) we have

$$\operatorname{arcosh}\left(1+\frac{2d_{\mathbb{E}}(O,P_k)^2}{1-d_{\mathbb{E}}(O,P_k)^2}\right)=\frac{1}{2}\operatorname{arcosh}\left(1+\frac{2d_{\mathbb{E}}(O,V_k)^2}{1-d_{\mathbb{E}}(O,V_k)^2}\right),$$

and by applying $\cosh(\cdot)$ to both sides, we get

$$1+\frac{2d_{\mathbb{E}}(O,P_k)^2}{1-d_{\mathbb{E}}(O,P_k)^2}=\cosh\left(\frac{1}{2}\operatorname{arcosh}\left(1+\frac{2d_{\mathbb{E}}(O,V_k)^2}{1-d_{\mathbb{E}}(O,V_k)^2}\right)\right). \tag{III.6}$$

We use the identity

$$\cosh\left(\frac{x}{2}\right)=\sqrt{\frac{\cosh(x)+1}{2}},$$

and from Equation (III.6) we get

$$1+\frac{2d_{\mathbb{E}}(O,P_k)^2}{1-d_{\mathbb{E}}(O,P_k)^2}=\sqrt{1+\frac{d_{\mathbb{E}}(O,V_k)^2}{1-d_{\mathbb{E}}(O,V_k)^2}}.$$

We substitute $d_{\mathbb{E}}(O,V_k)=2^{-1/4}$ and we obtain

$$1+\frac{2d_{\mathbb{E}}(O,P_k)^2}{1-d_{\mathbb{E}}(O,P_k)^2}=\sqrt{1+\frac{2^{-1/2}}{1-2^{-1/2}}}=\sqrt{1+\frac{\sqrt{2}}{2-\sqrt{2}}}. \tag{III.7}$$

Equation (III.7) has two opposite solutions, and since we are looking for a distance, we choose the positive solution; finally,

$$d_{\mathbb{E}}(O,P_k)=\frac{2^{1/4}}{\sqrt{2}+\sqrt{2-\sqrt{2}}}.$$

By consequence, P_0 is the rotation of the point $\left(\frac{2^{1/4}}{\sqrt{2}+\sqrt{2-\sqrt{2}}},0\right)$ by $-\frac{\pi}{8}$, and the result of this rotation is

$$P_0=\left(\frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}},-\frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right).$$

The other points P_k can be obtained by rotating P_0 by $\frac{k\pi}{4}$ for $k=1,\dots,7$. The results of our calculations for the dummy points are reported in the second column of Table III.7.

The exact coordinates obtained with the process described above would increase the algebraic degree of the predicates (studied in Section III.4) in an artificial way; therefore, we introduce a set $\mathcal{Q}_2'$ of rational approximations of the points in $\mathcal{Q}_2$. See the third column of Table III.7. These rational approximations have been obtained with MAPLE's function `convert(evalf(x), 'rational', digits)` with 3 digits for each algebraic expression. We have verified that $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2)$ and $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2')$ have identical combinatorial

structures with an *a posteriori* verification process: we initialize naively a Delaunay triangulation of $\mathbb{M}_2$ with the set $\mathcal{Q}_2$ and another one with the set $\mathcal{Q}_2'$ with the same combinatorics. We then execute an automatic validation process on each triangulation that verifies that it is a simplicial complex and the Delaunay property is satisfied for all triangles. The process gives no error in both cases, which indicates that $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2)$ and $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2')$ are both Delaunay triangulations of $\mathbb{M}_2$, while by construction their combinatorics are the same.

We initialize the triangulation of $\mathbb{M}_2$ as $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2')$. Note that any other set of points that satisfies condition (II.36) could be used to initialize the triangulation.

The triangulation, initialized with the set of dummy points $\mathcal{Q}_2'$, is shown in Figure III.8. This figure is obtained from the actual CGAL implementation that we will discuss into detail in Chapter IV.[8]

III.2.2 Finding faces in conflict with a new point

Let $p \in \mathcal{S} \subset \mathscr{D}_2$ be a new point to be inserted in the Delaunay triangulation. Consider $\boldsymbol{\sigma}$ a face in $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$, and recall definition (III.3) for the set Σ. We say that $\boldsymbol{\sigma}$ is *in conflict* with the input point p if there exists a face $\sigma \in \Sigma$ whose circumscribing disk contains p. The set of all faces in $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ that are in conflict with p is called the *conflict zone* of p.

Recall that, since hyperbolic circles are Euclidean circles, a Delaunay triangulation in $\mathbb{H}^2$ has exactly the same combinatorics as the Euclidean Delaunay triangulation of the same points, as long as all circumscribing circles are contained in the Poincaré disk. Consequently, the Euclidean Delaunay triangle containing p gives us a hyperbolic Delaunay face in conflict with p; the Euclidean and hyperbolic faces will both be denoted as σ_p, which should not introduce any confusion. To find this triangle, we adapt the so-called *visibility walk* [DPT02]. This walk starts from an arbitrary face, then, for each visited face, it visits one of its neighbors, until a face containing p is found. Before specifying how the neighbor to be visited is specified in the case of the Bolza surface, we introduce the notion of *neighbor translation*.

Definition III.8 (Neighbor translation)**.** Let $\boldsymbol{\sigma}, \boldsymbol{\tau}$ be two adjacent faces in $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ and σ, τ two of their preimages by π_2 in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{S})$. We define the neighbor translation $\mathrm{Ntr}(\sigma, \tau)$ from σ to τ as the translation of Γ_2 such that $\mathrm{Ntr}(\sigma, \tau)\tau$ is adjacent to σ in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{S})$.

Let v be a vertex common to $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$, and let v_σ and v_τ the vertices of σ and τ that project on v by π_2. We can compute the neighbor translation from σ to τ as

$$\mathrm{Ntr}(\sigma, \tau) = \mathrm{T}(v_\tau, \tau)\, \mathrm{T}^{-1}(v_\sigma, \sigma).$$

[8] Note that screenshots of the implementation were **not** included in [IT17].

Point	Expression	Rational approximation
V_0	$\left(\frac{2^{3/4}\sqrt{2+\sqrt{2}}}{4}, -\frac{2^{3/4}\sqrt{2-\sqrt{2}}}{4}\right)$	$(97/125, -26/81)$
M_4	$\left(-\sqrt{\sqrt{2}-1}, 0\right)$	$(-9/14, 0)$
M_5	$\left(-\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}, -\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}\right)$	$(-5/11, -5/11)$
M_6	$\left(0, -\sqrt{\sqrt{2}-1}\right)$	$(0, -9/14)$
M_7	$\left(\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}, -\frac{\sqrt{2}\sqrt{\sqrt{2}-1}}{2}\right)$	$(5/11, -5/11)$
P_0	$\left(\frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, -\frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(1/2, -4/19)$
P_1	$\left(\frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}+\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, \frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}-\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(1/2, 4/19)$
P_2	$\left(\frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, \frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(4/19, 1/2)$
P_3	$\left(\frac{2^{3/4}\left(\sqrt{2-\sqrt{2}}-\sqrt{2+\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, \frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}+\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(-4/19, 1/2)$
P_4	$\left(-\frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, \frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(-1/2, 4/19)$
P_5	$\left(-\frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}+\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, \frac{2^{3/4}\left(\sqrt{2-\sqrt{2}}-\sqrt{2+\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(-1/2, -4/19)$
P_6	$\left(-\frac{2^{1/4}\sqrt{2-\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}, -\frac{2^{1/4}\sqrt{2+\sqrt{2}}}{2\sqrt{2}+2\sqrt{2-\sqrt{2}}}\right)$	$(-4/19, -1/2)$
P_7	$\left(\frac{2^{3/4}\left(\sqrt{2+\sqrt{2}}-\sqrt{2-\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}, -\frac{2^{3/4}\left(\sqrt{2-\sqrt{2}}+\sqrt{2+\sqrt{2}}\right)}{4\sqrt{2}+4\sqrt{2-\sqrt{2}}}\right)$	$(4/19, -1/2)$

Table III.7: Exact and rational expressions for the dummy points

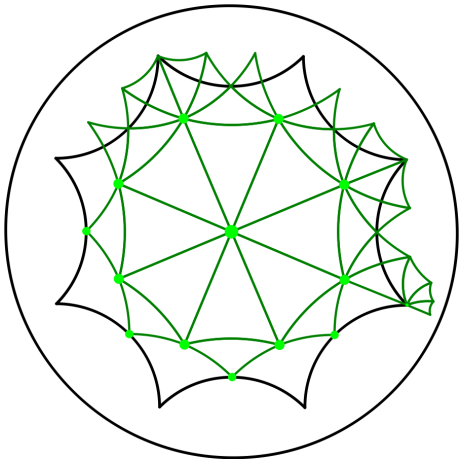

Figure III.8: Canonical representatives of the Delaunay triangulation of the Bolza surface defined by the set $\mathcal{Q}_2'$ of rational dummy points. This image is a screenshot of the actual CGAL *implementation.*

Figure III.9 illustrates the neighbor translation of the canonical representatives of $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. It can be easily seen that

$$\mathrm{Ntr}(\sigma,\tau) = \mathrm{T}(\nu,\tau)\ \mathrm{T}^{-1}(\nu,\sigma) = \Big(\mathrm{T}(\nu,\sigma)\ \mathrm{T}^{-1}(\nu,\tau)\Big)^{-1} = \mathrm{Ntr}^{-1}(\tau,\sigma).$$

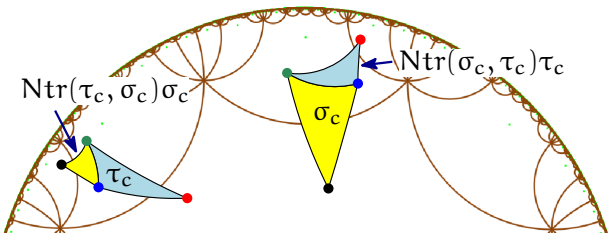

Figure III.9: Translating τ^c by $\mathrm{Ntr}(\sigma^c,\tau^c)$ gives a face adjacent to σ^c.

We define the *location translation* L_{tr} as follows: let σ_p be the Euclidean Delaunay triangle containing p. L_{tr} is the translation that moves σ_p^c to σ_p.

The location procedure starts from a face incident to O. Then, for each visited face σ of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$, we consider the Euclidean edge e defined by two of the vertices of σ. With a simple orientation test, we can check whether the Euclidean line supporting e separates p from the vertex of σ opposite to e. If this is the case, the next visited face is the neighbor τ of σ through e, and we repeat the process, until

- either we find the Euclidean Delaunay face σ_p containing p by visiting only faces that do not cross the border of D_2; then σ_p is a (canonical) face of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$ in conflict with p, and $L_{tr} = \mathrm{Id}$.

- or, at some point, we visit a (canonical) face σ_{D_2} included in D_2 and its (non-canonical) neighbor τ that crosses the border of D_2. Then the walk continues in non-canonical faces, until we find the Euclidean triangle σ_p containing p. Then L_{tr} is $\mathrm{Ntr}(\sigma_{D_2}, \tau^c)$ and the canonical face in conflict with p is $\sigma_p^c = L_{tr}^{-1}\sigma_p$.

If a (Euclidean) face with edges e_1, e_2, and e_3 is entered through e_1 during the walk, and if none of e_2 and e_3 separates its opposite vertex from p, then the face contains p. So, two orientation tests are enough to conclude that a face contains p (except for the starting face).

The location translation L_{tr} is also used when looking for the conflict zone of p. Starting from σ_p^c and for each face in conflict with p, we recursively examine the translated image under L_{tr} of each neighbor (obtained with a neighbor translation) that has not yet been visited. We store the set Z^c of canonical faces in conflict with p. Note that Z^c is not necessarily a connected region.

Examples of the conflict zone of an input point can be seen in Figure III.10. Note the last example, in which the conflict zone consists of canonical representatives that are not adjacent in $\mathbb{H}^2$.

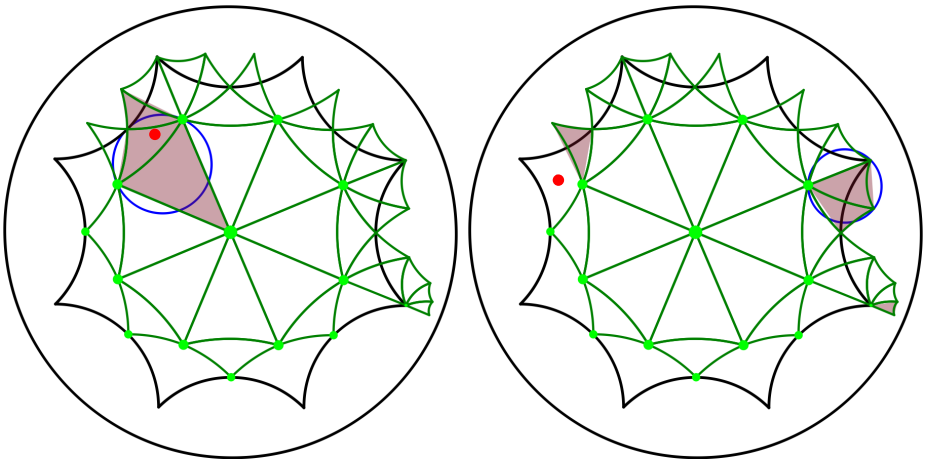

Figure III.10: Two examples showing the conflict zone of the red point point in $\mathscr{D}_2$. Note that in the example on the right the conflict zone is not a connected region in $\mathbb{H}^2$.

III.2.3 Insertion

Once we have obtained the conflict zone of a point p, we may proceed with its insertion in the triangulation. This process consists in creating new faces with p as an apex, and deleting the faces in conflict with p. The location translation L_{tr} introduced in Section III.2.2 will again be used. We know that p lies in the Euclidean triangle $L_{tr}\sigma_p^c$. We first create a new vertex v_{new} and store p in it.

By construction, the union of all translated faces

$$L_{tr}Ntr\big(\sigma_p^c, \tau^c\big)\tau^c, \qquad \tau^c \in Z^c$$

is a topological disk Z in $\mathbb{H}^2$.

We identify the sequence of edges E on the border of Z; each edge e is incident to one face in Z and one face that is not in Z. For each face τ^c in Z^c, we temporarily store the translations $L_{tr}Ntr(\sigma^c, \tau^c)T(v_i, \tau^c), i = 0, 1, 2$ directly in its three vertices (not in $\boldsymbol{\tau}$, since it will be deleted). Since Z is a topological disk, the result for a given vertex v is independent of the face of Z incident to v that is considered. We store Id in vertex v_{new}.

For each edge $e \in E$, we create a new face $\boldsymbol{\tau}_e$ having e as an edge and v_{new} as third vertex. The neighbor of $\boldsymbol{\tau}_e$ outside Z^c is the neighbor through e of the face in Z^c incident to e. Two new faces consecutive along E are adjacent. We can now delete all faces in Z.

All that is left to do now is to compute the translations to be stored in the new faces. Let τ_{new} be a newly created face. We retrieve the translations temporarily stored in its vertices v_0, v_1, v_2 and we store them in τ_{new}. Equipped with these translations, τ_{new} is not necessarily canonical. If all translations stored in τ_{new} are equal to Id, then τ_{new} is contained in $\mathscr{D}_2$, so it is actually canonical. Otherwise, one of the vertices of τ_{new} is v_{new}; without loss of generality, $v_0 = v_{new}$, and $T(v_0, \tau_{new}) = Id$. For $i = 0, 1, 2$ we can easily compute $d_i = idx_{\mathcal{N}_2}\big(T(v_{\tau_i}^{out}, \tau_i)\big)$, where τ_i is the image of τ_{new} under $T^{-1}(v_i, \tau_{new})$: in each face, $v_{\tau_i}^{out}$ is the first vertex of τ_i such that $T(v_{\tau_i}^{out}, \tau_i) \neq Id$. Note that we do not actually compute the images of τ_{new}, we only compute translations (as words). We then find the index k for which d_k is minimal, and in $\boldsymbol{\tau}_{new}$ we store the translations $T^{-1}(v_k, \tau_{new})T(v_i, \tau_{new})$, $i = 0, 1, 2$. The face τ_{new} has now been canonicalized. Once this is done for all new faces, temporary translations can be removed from the vertices.

An example of the insertion of a single new point in the triangulation is shown in Figure III.11.

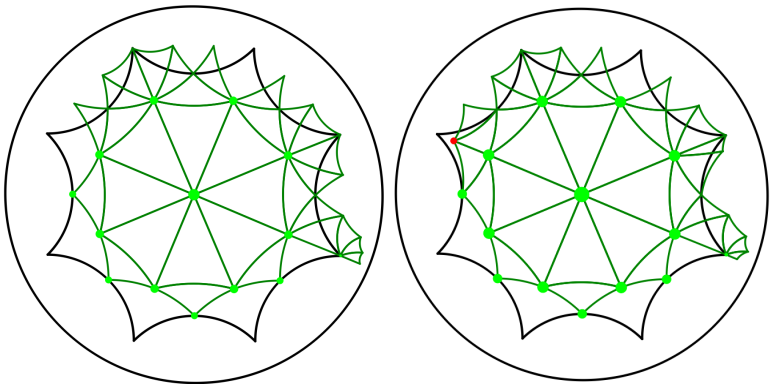

Figure III.11: Example of the insertion of a point (right) in the triangulation of $\mathbb{M}_2$ defined by the dummy points (left).

III.3 Operations on a triangulation

In this section we describe the location of a point in the triangulation, as well as the removal of existing vertices. This will allow us also to discuss the removal of dummy points from a triangulation of the Bolza surface, which is the last step of the incremental algorithm presented in Section II.3.2. Note that the operations presented in this section were not described in [IT17].

III.3.1 Point location

In Section III.2.2 we described the visibility walk, which allows to identify a face $\boldsymbol{\sigma}_p$ in conflict with a point p. As we saw, in fact, the image of σ_p^c under the location translation L_{tr} is a face σ_p whose vertices define a Euclidean triangle that contains p. In this section we discuss a way to identify the *hyperbolic* triangle that contains the point p, as it may not necessarily be the one defined by the vertices of σ_p. See Figure III.12. Note that hyperbolic location was not presented in [IT17].

In order to check if p is contained in the hyperbolic triangle defined by the vertices of σ_p, we proceed as follows. Each hyperbolic edge e_k, $k = 0, 1, 2$, is realized in $\mathbb{H}^2$ as a hyperbolic segment, supported either by a Euclidean circle or by a Euclidean line. If, for each vertex v_k of σ_p, v_k and p are both on the same side of the supporting circle (or line) of e_k, then p is contained in the hyperbolic triangle defined by the vertices of σ_p, and so p lies in the hyperbolic triangle $L_{tr}\sigma_p^c$. In this case p is located in $\boldsymbol{\sigma}_p$ with location translation L_{tr}.

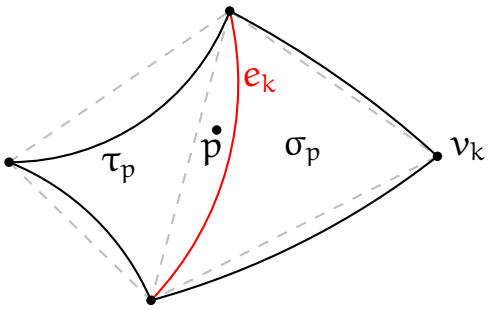

Figure III.12: Point location in the hyperbolic plane. The point p lies in the Euclidean triangle σ_p, but not in the hyperbolic triangle with the same vertices. Since p and v_k are on different sides of the edge e_k, p lies in the hyperbolic triangle τ_p, which is the neighbor of σ_p opposite to v_k.

If the test fails for some edge e_k of σ_p, then p is contained in the neighbor of σ_p through e_k, which we denote as τ_p. In this case, p is located in $\boldsymbol{\tau}_p$ with location translation $L'_{tr} = L_{tr} * \mathrm{Ntr}(\sigma_p^c, \tau_p^c)$; so, p lies in $L'_{tr}\tau_p^c$.

The hyperbolic location is illustrated in the screenshots shown in Figure III.13.

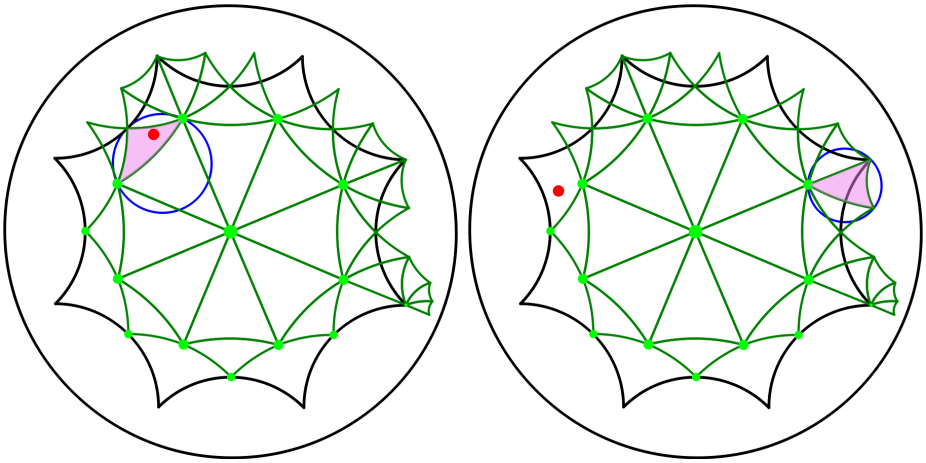

Figure III.13: Examples of hyperbolic location. The circle circumscribes the canonical face in which the red point is located. Note how in the example on the right the point is located in a face whose canonical representative is on the other side of the octagon.

III.3.2 Removal of an existing vertex

In this section we discuss a functionality that was not presented in [IT17]. Vertex removal is the last piece that makes the implementation fully dynamic.

Let $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ be the Delaunay triangulation of $\mathbb{M}_2$ defined by a set of points $\mathcal{S}$ that satisfies the validity condition (II.36). We want to remove a vertex v that stores a point $p \in \mathcal{S}$ from the triangulation. The removal of v consists in computing $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S} \setminus \{p\})$. Clearly, a constraint is imposed on the removal of vertices due to the validity condition: v can be removed from $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ only if the set $\mathcal{S} \setminus \{p\}$ satisfies Equation (II.36).

There exist multiple algorithms for the removal of vertices from a Delaunay triangulation [AGSS89, Che90, Dev02, DT03]. We will be interested in a well-known algorithm that is sub-optimal, but fairly simple to conceive and implement, and also quite efficient. The algorithm that we discuss has already been used by Teillaud for the construction of Delaunay triangulations in 3 dimensions [JPT14] and by Caroli for triangulations of the flat torus in 3 dimensions [Car10], and it is used in the current implementation in CGAL.

Given a set of points $\mathcal{P}$ in $\mathscr{D}_2$, the combinatorial structure of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{P})$ and the combinatorial structure of $\mathrm{DT}_{\mathbb{E}^2}(\Gamma_2\mathcal{P})$ are identical, since the circumscribing disks of all faces in the triangulation are included in the Poincaré disk and Euclidean and hyperbolic disks in $\mathbb{H}^2$ coincide. The idea behind the algorithm is to compute the Euclidean Delaunay triangulation of the vertices adjacent to v, to "sew" this triangulation combinatorially into the hyperbolic triangulation, and finally to destroy v and its incident faces. With this idea in mind, we describe now the steps of the algorithm that we use to remove v from $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$.

As mentioned in Section III.1.2, each vertex in the triangulation data structure that

we use gives access to one of its incident faces. Let $\boldsymbol{\sigma}_v$ be the face to which v points. Starting from σ_v^c, we move from neighbor to neighbor counter-clockwise around v using adjacency relations to obtain the set of canonical faces F_v^c incident to v. Note that, similarly to the conflict zone of an input point, the set F_v^c is not necessarily a connected region in $\mathbb{H}^2$. In order to obtain images of the faces in F_v^c that form a connected region in $\mathbb{H}^2$, for each face τ_v^c in F_v^c we apply the neighbor translation $\mathrm{Ntr}(\sigma_v^c, \tau_v^c)$ to τ_v^c. The resulting set F_v of translated faces is a topological disk in $\mathbb{H}^2$.

We recover the vertices on the boundary of F_v, i.e., the set of vertices of the faces in F_v that are not v. Recall that each boundary vertex v_k of each face τ_v^c in F_v has been translated by $T_{v_k} = \mathrm{Ntr}(\sigma_v^c, \tau_v^c)T(v_k, \tau_v^c)$ during the creation of the set F_v. Each translation T_{v_k} is temporarily stored in each vertex v_k. We recover also the set of boundary edges e_k, i.e., the set of edges incident to a face in F_v and to a face not in F_v. Both the set of boundary vertices and edges are essential to the sewing step.

From the set of boundary vertices, we compute the set of boundary points $\mathcal{P}_v$ by applying the translation T_{v_k} stored in each vertex v_k to the point p_k it stores. We then compute the Euclidean Delaunay triangulation of the points in $\mathcal{P}_v$. The insertion of each point of $\mathcal{P}_v$ yields a vertex of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$ that corresponds to one of the boundary vertices v_k. This correspondence allows us to actually "sew" $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$ into $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$.

It should be noted that the Euclidean Delaunay triangulation of $\mathcal{P}_v$ gives the triangulation of the *convex hull* of $\mathcal{P}_v$ in the unit disk, while the union of the faces in F_v is not necessarily convex. "Unnecessary" faces of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$ can be identified by comparing the orientation of their boundary vertices with the orientation of the boundary vertices of the union of F_v. A face in $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$ whose orientation of the boundary vertices disagrees with the orientation of the boundary vertices of F_v is considered unnecessary, and not treated from this point onwards.

Before we proceed to actually modify $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$, we must verify that its removal will not violate the validity condition (II.36). Note that this check is made only at this point because we need to know the combinatorial structure of the re-triangulated region, and for this we need $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$. Since the removal of v only replaces the faces in F_v with new faces, we need to check only locally that none of the faces of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$ is too large. In other words, we need to verify that for each face τ of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$

$$\max \delta(\tau) < \tfrac{1}{2}\,\mathrm{sys}(\mathbb{M}_2), \tag{III.9}$$

where and $\delta(\tau)$ is the *hyperbolic* diameter of the disk circumscribing τ. If (III.9) is satisfied, then the removal of v is allowed; otherwise, the removal is aborted.

Next is the sewing step. We create new hyperbolic faces with the same combinatorial structure as $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$. The correspondence mapping between vertices of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$ and the boundary vertices v_k of F_v makes the creation of the new faces straightforward. Adjacency relations between the new faces are set progressively as they are created, using the combinatorial structure of $\mathrm{DT}_{\mathbb{E}^2}(\mathcal{P}_v)$. To set adjacency relations with the boundary faces of $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$, we use the information provided by the boundary edges e_k.

Once all adjacency relations have been set, the faces in F_v are destroyed, along with the vertex v.

The final step is to canonicalize the new faces as described at the end of Section III.2.3. The translations stored in the boundary vertices are restored in the new faces, and each face is canonicalized. At the end of the canonicalization process, the translations stored in the vertices are removed.

The removal of vertices is illustrated via the removal of dummy points shown in the screenshots in Figure III.14.

III.3.3 Removal of dummy points

The removal of dummy points is also new with respect to [IT17].

The set of dummy points $\mathcal{Q}_2$ is used to initially satisfy the validity condition (II.36). However, the dummy points are artificial, and their presence in the final triangulation might be undesirable or even problematic for users. For this reason, with the insertion of new points in the triangulation, we try to remove as many dummy points as possible, without violating the validity condition.

Let $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2')$ be the triangulation of the Bolza surface defined by the set of dummy points, and let $\mathcal{P}$ be a set of input points to be inserted into $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2')$. We consider two strategies for the removal of dummy points:

a) **greedy:** after the insertion of a single point $p \in \mathcal{P}$, we try to remove, one by one, the dummy points currently in the triangulation of $\mathbb{M}_2$.[9]
b) **lazy:** the whole set of points $\mathcal{P}$ is inserted in the triangulation, and in the end we try to remove the dummy points one by one.

The insertion and removal of points are done as described in Sections III.2.3 and III.3.2. If the set of input points $\mathcal{P}$ is sufficiently large and well-distributed, then all dummy points are removed from the triangulation. If this is not the case, the triangulation will contain the dummy points that could not have been removed due to condition (II.36). Note that, for the same set of input points $\mathcal{P}$, both strategies give the same final triangulation. An example of this functionality is shown in Figure III.14.

III.4 Algebraic complexity

The combinatorial validity of Delaunay triangulations in CGAL relies on the exact computation of *predicates*, which will be discussed in detail in Section IV.1. At this point it suffices to say that a predicate is a question that has either affirmative or negative

[9] This process can be optimized by trying to remove only the dummy points "close enough" to p, where "close enough" could be defined either in a geometric sense (i.e., the dummy points within a certain distance to p) or in combinatorial sense (i.e., the dummy points adjacent to p or adjacent to the neighbors of p). In the current implementation, no such optimizations are done.

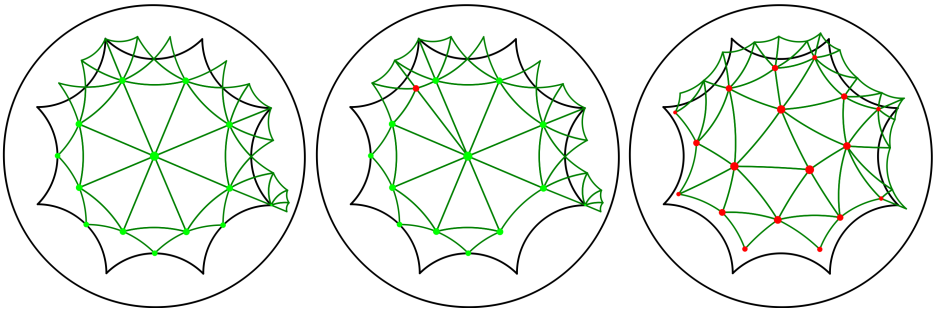

Figure III.14: As points are inserted in the triangulation, dummy points are removed. In the second screenshot (middle), already with the insertion of one new point, a dummy point has been removed. In the last picture (right), no dummy point is present in the triangulation.

answer. In this section we present the predicates needed for our computations, and in the sequel we review their algebraic complexity.

III.4.1 Predicates

As can be seen in Sections III.2 and III.3, the correctness of the combinatorial structure $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ relies on the exact evaluation of four predicates:

- SideOfOriginalOctagon, which checks whether an input point lies inside the half-open original domain $\mathscr{D}_2$. This predicate is used as a precondition for the insertion of each point.
- Orientation, which checks whether an input point p in $\mathscr{D}_2$ lies on the right side, the left side, or on an oriented Euclidean segment. This predicate is used when looking for the Euclidean triangle containing an input point.
- InCircle, which checks whether an input point p in $\mathscr{D}_2$ lies inside, outside, or on the boundary of the disk circumscribing an oriented triangle. It is used when looking for all faces in conflict with an input point.
- SideOfOrientedHyperbolicSegment, which checks whether an input point p lies on the right side, on the left side, or on the hyperbolic line that supports an oriented hyperbolic segment. It is used when looking for the hyperbolic triangle that contains p.

Let the coordinates of a point $p_i \in \mathbb{H}^2$ be denoted as x_i and y_i. The Orientation and InCircle predicates can be expressed as signs of determinants:

$$\textsc{Orientation}(p_1, p_2, p_3) = \operatorname{sign} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, \tag{III.10}$$

$$\textsc{InCircle}(p_1, p_2, p_3, p_4) = \operatorname{sign} \begin{vmatrix} x_1 & y_1 & x_1^2 + y_1^2 & 1 \\ x_2 & y_2 & x_2^2 + y_2^2 & 1 \\ x_3 & y_3 & x_3^2 + y_3^2 & 1 \\ x_4 & y_4 & x_4^2 + y_4^2 & 1 \end{vmatrix}. \tag{III.11}$$

Note that the InCircle predicate is evaluated as in (III.11) if the orientation of the points p_1, p_2, and p_3 is positive.

The SideOfOriginalOctagon predicate is evaluated via the Orientation and InCircle predicates. The evaluation of SideOfOriginalOctagon is done in two steps: a preliminary check whether the point lies inside the closed octagon D_2 or not, and a more specific check if the point is located on the boundary of D_2, in which case it must be verified whether the point is on one of the closed sides of the half-open octagon $\mathscr{D}_2$ or if it is the vertex V_0.

A naive approach to checking whether a query point p lies inside D_2 is to verify that p lies outside of all the Euclidean circles that support the eight sides of D_2. However, this approach would involve 8 InCircle tests, which is something we would like to avoid. We exploit the rotational symmetry of D_2 and, instead of checking if p lies in D_2, we check if the rotational image p' of p in the first (closed) octant lies in the intersection of D_2 and the octant shown in Figure III.15.

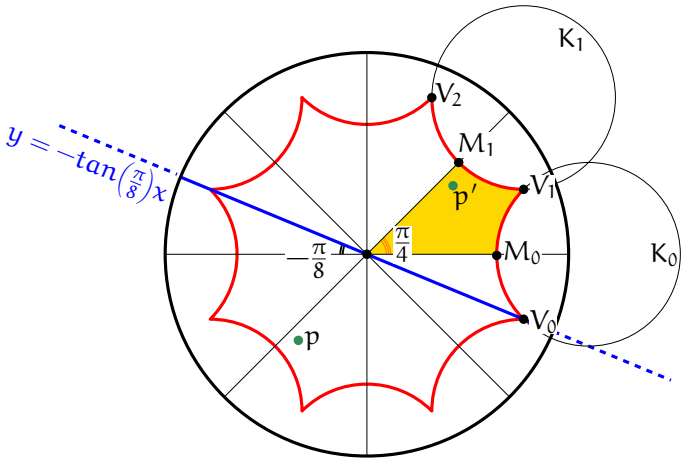

Figure III.15: Illustration of the SideOfOriginalOctagon *predicate. Instead of checking if a point p lies in the octagon, we check instead if its rotational image in the first octant lies in the shaded zone.*

In order to find the image p' of p in the first octant, we first take the absolute value

of the coordinates of p, and then, if the y-coordinate of the result is larger than its x-coordinate, we swap them.

Now the SideOfOriginalOctagon test boils down to two InCircle tests, involving the Euclidean circles K_0 and K_1 shown in Figure III.15. Note that these Euclidean circles can be recovered from the work of Balazs and Voros [BV86, Fig. 16], [10] however we do not need the actual equations of the circles. For the evaluation of the InCircle predicate, we just need three points lying on each circle, and we already have this information from Table III.7. K_0 passes through the points V_0, M_0 and V_1, while K_1 passes through V_1, M_1 and V_2. The only variable for the SideOfOriginalOctagon predicate is the point p.

If the point p' lies inside either K_0 or K_1, then p is outside D_2, and by extension, outside $\mathscr{D}_2$. If p' is outside both Euclidean circles, then p is inside D_2, and by extension inside $\mathscr{D}_2$. If none of these cases is verified, then p lies on the boundary of the closed octagon D_2, and we need to take cases to check if p is a point of the half-open octagon $\mathscr{D}_2$ or not. If p' lies on both K_0 and K_1, then p is a vertex of the closed octagon D_2; if $p = V_0$, then p is inside $\mathscr{D}_2$, otherwise it is outside. Finally, if p' lies only on K_0 or on K_1, then p lies on a side of D_2, and we must verify whether it is an open side of $\mathscr{D}_2$ or not. If p lies beneath the Euclidean line $y = -\tan(\frac{\pi}{8})x$ (shown in Figure III.15), then p is inside $\mathscr{D}_2$; otherwise, p is outside $\mathscr{D}_2$.

Finally, we discuss how the predicate SideOfOrientedHyperbolicSegment is evaluated. Note that this predicate was not described in [IT17]. The predicate SideOfOrientedHyperbolicSegment is evaluated via one call either to the Orientation or to the InCircle predicate. The predicate accepts three points $p_i, i = 0, 1, 2$ in $\mathbb{H}^2$ as arguments, and returns the relative position of p_2 with respect to the oriented hyperbolic segment with endpoints p_0 and p_1. An oriented hyperbolic segment is supported either by an oriented Euclidean line or by an oriented Euclidean circle. Depending on the case, we use the appropriate predicate.

III.4.2 Algebraic degree of predicates

We assume that all input points (which lie in $\mathscr{D}_2$) have rational coordinates. Recall that this holds for the initial dummy points, see Section III.2.1. So, in determinants (III.10) and (III.11), at least one point (x_i, y_i) is rational. However, the points against which the predicates are testing the new input point are vertices of some face of $DT_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$ contained in $\mathcal{U}_{D_2}$, so they are images of some input points by translations in $\mathcal{N}_2$. Therefore, the evaluation of the two predicates (III.10) and (III.11) boils down to determining the sign, considered as an element of $\{-1, 0, 1\}$, of polynomial expressions in rational

[10] For the sake of completeness, the centers of these circles lie at Euclidean distance $\sqrt{\frac{1}{2}\left(\sqrt{2}+1\right)}$ from the origin, and their radii are equal to $\sqrt{\frac{1}{2}\left(\sqrt{2}-1\right)}$.

variables whose coefficients are lying in some extension field of the rationals, as made precise below.

The evaluation of the degree of the predicates requires to perform a case analysis on the different possible positions of the faces in $\mathcal{U}_{D_2}$, i.e., on the possible translations of $\mathcal{N}_2$ that can be involved in each predicate. The following property shows how we can take symmetries of D_2 into account to reduce the number of possible cases.

Lemma III.12. *Let $\boldsymbol{\sigma}$ be a face in $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$. Then, for any edge uv of its canonical representative σ^c, such that $T(u,\sigma^c) \neq \mathrm{Id}$ and $T(v,\sigma^c) \neq \mathrm{Id}$,*

$$\left| idx_{\mathcal{N}_2}(T(u,\sigma^c)) - idx_{\mathcal{N}_2}(T(v,\sigma^c)) \right| \leq 7.$$

Proof. We can assume that $\sigma^c \not\subset \mathscr{D}_2$, otherwise all its three translations are equal to Id. Reusing the proof of of Lemma III.1 and the notation therein, we see that σ^c is either contained in the disk bounded by B_k, or in the disk bounded by C_k, for some $k \in \{0,\dots,7\}$. So σ^c can only intersect D_2 and the seven octagons around some V_k. The result follows. □

Recall Figure III.3 - Right, which shows the possible translations of $\mathcal{N}_2$ involved in a given canonical representative, for some $k \in \{0,\dots,7\}$. Their matrices are given in Table III.16.

For k even, the sine and cosine of $\pm k\pi/4$ have values in $\{-1,0,1\}$, while for k odd they are both equal to $\pm\sqrt{2}/2$. Therefore, up to sign, the above matrices are divided into two "classes". Due to the symmetries of D_2, we actually only need to examine one case in each class, therefore we can focus on the two cases $k = 0$ and $k = 1$.

In [IT17] the maximum degree of the predicates was given in a single proposition and the proofs were incomplete. Since there is no space limitation here, we will examine the degree of each predicate individually.

Proposition III.13. *The* Orientation *predicate can be evaluated by determining the sign of rational polynomial expressions of total degree at most 20 in the coordinates of input points.*

Proof. As mentioned above, at least one point is inside $\mathscr{D}_2$. Without loss of generality, we assume that $p_3 \in \mathscr{D}_2$. Let us consider the possible cases for the other two points.

- All three points are inside $\mathscr{D}_2$. In this case, all the arguments of the predicate are rational, so from (III.10) we get a polynomial with rational coefficients of total degree 2 in the coordinates of the input points.
- Point p_2 is also in $\mathscr{D}_2$, and p_1 is outside $\mathscr{D}_2$. In this case, p_1 can be the image of an input point by 14 possible different translations in $\mathcal{N}_2$ (seven around V_0 and seven around V_1).

$$
\begin{aligned}
T_k &= \begin{bmatrix} 1+\sqrt{2} & e^{ik\pi/4}\sqrt{2}\xi \\ e^{-ik\pi/4}\sqrt{2}\xi & 1+\sqrt{2} \end{bmatrix} \\
T_k T_{k+3} &= \begin{bmatrix} \left(1+\sqrt{2}\right)\left(1-i\sqrt{2}\right) & e^{ik\pi/4}\left(1+i\left(1+\sqrt{2}\right)\right)\xi \\ e^{-ik\pi/4}\left(1-i\left(1+\sqrt{2}\right)\right)\xi & \left(1+\sqrt{2}\right)\left(1+i\sqrt{2}\right) \end{bmatrix} \\
T_k T_{k+3} T_{k+6} &= \begin{bmatrix} -\left(1+\sqrt{2}\right)(1+2i) & e^{ik\pi/4}\left(1+\sqrt{2}\right)(-1+i)\xi \\ -e^{-ik\pi/4}\left(1+\sqrt{2}\right)(1+i)\xi & -\left(1+\sqrt{2}\right)(1-2i) \end{bmatrix} \\
T_k T_{k+3} T_{k+6} T_{k+1} &= \begin{bmatrix} -2\sqrt{2}-3 & -e^{ik\pi/4}\left(2+\sqrt{2}+i\sqrt{2}\right)\xi \\ -e^{-ik\pi/4}\left(2+\sqrt{2}-i\sqrt{2}\right)\xi & -2\sqrt{2}-3 \end{bmatrix} \\
T_{k+1} T_{k+6} T_{k+3} &= \begin{bmatrix} \left(1+\sqrt{2}\right)(-1+2i) & -e^{ik\pi/4}\left(2+\sqrt{2}\right)i\xi \\ e^{-ik\pi/4}\left(2+\sqrt{2}\right)i\xi & -\left(1+\sqrt{2}\right)(1+2i) \end{bmatrix} \\
T_{k+1} T_{k+6} &= \begin{bmatrix} \left(1+\sqrt{2}\right)+\left(2+\sqrt{2}\right)i & e^{ik\pi/4}\left(1+\sqrt{2}-i\right)\xi \\ e^{-ik\pi/4}\left(1+\sqrt{2}+i\right)\xi & \left(1+\sqrt{2}\right)-\left(2+\sqrt{2}\right)i \end{bmatrix} \\
T_{k+1} &= \begin{bmatrix} 1+\sqrt{2} & e^{ik\pi/4}(1+i)\xi \\ e^{-ik\pi/4}(1-i)\xi & 1+\sqrt{2} \end{bmatrix}
\end{aligned}
$$

Table III.16: Matrices of translations around a vertex V_k, k. *Here,* $\xi = \sqrt{1+\sqrt{2}}$.

- Only p_3 is inside $\mathscr{D}_2$. In this case, both p_1 and p_2 can be images of input points under the translations around V_0 and V_1. Of course, we avoid redundancies: if we examine the case ORIENTATION $(T_i p_1', T_j p_2', p_3)$, $p_i', p_j' \in \mathscr{D}_2$, we do not examine the case ORIENTATION $(T_j p_1', T_i p_2', p_3)$ since it would have the same degree. This amounts to 56 cases in total—28 cases around V_0 and another 28 around V_1.

We have found with Maple[11] that in all cases, the expressions produced by (III.10) have denominators that are strictly positive and numerators that can be brought into the form

$$\left(A\sqrt{2} + B\right)\xi + C\sqrt{2} + D, \qquad \xi = \sqrt{1+\sqrt{2}}, \tag{III.14}$$

where A, B, C, D are rational polynomial expressions in the input coordinates. Moreover, the maximum total degree of A, B, C, D is 5. By squaring twice (to eliminate square roots coming from ξ), we get a rational polynomial of degree 20 in rational variables, which concludes the proof. □

Note that the proof for Proposition III.13 was given also in [IT17], but Propositions III.15 and III.16 not explicitly stated or proven in the paper.

Proposition III.15. *The* INCIRCLE *predicate can be evaluated by determining the sign of rational polynomial expressions of total degree at most 72 in the coordinates of input points.*

Proof. Similarly to the ORIENTATION predicate, we assume without loss of generality that p_4 is always inside $\mathscr{D}_2$, so we need to examine cases for the other three points.

- All points are inside $\mathscr{D}_2$. In this case, all the arguments of the predicate are rational and from Equation (III.11) we get a rational polynomial expression of total degree 4.
- Three points are inside $\mathscr{D}_2$—let $p_2, p_3, p_4 \in \mathscr{D}_2$, and suppose p_1 is outside $\mathscr{D}_2$. As for the ORIENTATION predicate, we have a total of 14 cases.
- Two points are inside $\mathscr{D}_2$—we consider $p_3, p_4 \in \mathscr{D}_2$. Both p_1 and p_2 can be images of input points under translations around V_0 and V_1, so we get 56 cases.
- Only p_4 is inside $\mathscr{D}_2$—here p_1, p_2 and p_3 can be images of input points under all translations around V_0 and V_1. Again avoiding redundancies, the total number of cases is 168 (84 combinations around V_0 and another 84 around V_1).

Similarly to the ORIENTATION predicate, in all listed cases the expressions resulting from Equation (III.11) have strictly positive denominators and their numerators can be brought into the form (III.14). With MAPLE, we find that the maximum total degree of the expressions A, B, C, D is 18. By squaring twice to eliminate square roots, we get that the maximum degree of the INCIRCLE predicate is 72. □

[11] The MAPLE worksheets for all proofs are available online at the address https://members.loria.fr/Monique.Teillaud/DT_Bolza_SoCG17/.

Proposition III.16. *The* SIDEOFORIGINALOCTAGON *predicate can be evaluated by determining the sign of rational polynomial expressions of total degree at most 8 in the coordinates of input points.*

Proof. The SIDEOFORIGINALOCTAGON predicate is much simpler than the previous two, as it only takes one point p as argument. As we have seen in Section III.4.1, the SIDEOFORIGINALOCTAGON predicate is evaluated with two INCIRCLE predicates. Once the INCIRCLE predicate is called with the input point p and the vertices V_0, M_0 and V_1 of D_2, and the other time with input p and the vertices V_1, M_1 and V_2. Our results in MAPLE allow us to conclude that the maximum degree for these specific cases is 2, and by squaring twice we get algebraic expressions of total degree 8 in the input coordinates. □

Proposition III.17. *The* SIDEOFORIENTEDHYPERBOLICSEGMENT *predicate can be evaluated by determining the sign of rational polynomial expressions of total degree at most 40 in the coordinates of input points.*

Proof. The SIDEOFORIENTEDHYPERBOLICSEGMENT predicate is evaluated with a call either to the ORIENTATION or the INCIRCLE predicate. As mentioned in Section III.4.1, the input to the predicate are three points $p_i, i = 0, 1, 2$. We assume that the point p_2 is always inside $\mathscr{D}_2$. Let us consider the two cases in which the oriented hyperbolic segment $[p_0, p_1]$ is supported either by a Euclidean line or by a Euclidean circle:

- The segment $[p_0, p_1]$ is supported by a Euclidean line. In this case, we use the ORIENTATION predicate to evaluate SIDEOFORIENTEDHYPERBOLICSEGMENT. Both points p_0 and p_1 may lie outside $\mathscr{D}_2$. According to the analysis reported in the proof of Proposition III.13, the maximum degree of SIDEOFORIENTEDHYPERBOLICSEGMENT in this case is 20.
- The segment $[p_0, p_1]$ is supported by a Euclidean circle. In this case, we need an additional point to define the Euclidean circle supporting the hyperbolic segment through p_0 and p_1. The points p_0 and p_1 may lie outside $\mathscr{D}_2$, but the auxiliary point can be chosen so that it lies in $\mathscr{D}_2$. From our computations in MAPLE we find that with two points inside $\mathscr{D}_2$ the maximum degree of A, B, C, D in Equation (III.14) is 10. Squaring twice to eliminate nested square roots gives a total algebraic degree 40.

□

Finally, from Propositions III.13 and III.15 to III.17, we give the following proposition.

Proposition III.18. *All predicates can be evaluated by determining the sign of rational polynomial expressions of total degree at most 72 in the coordinates of input points.*

Chapter IV

Implementation in CGAL

In the previous chapter, we discussed the details of the incremental algorithm for the construction of Delaunay triangulations of the Bolza surface. In this chapter, we discuss the actual implementation of the algorithm in CGAL.

We begin our discussion with a short introduction to CGAL. We continue with a brief presentation of the existing triangulation packages and give details about the components needed to construct a triangulation (geometric traits, kernels, triangulation data structure, face and vertex base classes). Next, we describe the classes that we introduce to the library and their connection to existing objects. We conclude with a presentation of our contribution to CGAL and give experimental results from our implementation.

Note that, while the larger part of the material discussed in Chapter III has been published in [IT17], the majority of the material in this chapter is not published.

Recall that triangulations of the Bolza surface can be seen as periodic Delaunay triangulations in $\mathbb{H}^2$. Our implementation [IT] depends logically and hierarchically on Delaunay triangulations of $\mathbb{H}^2$ [BIT]. Hyperbolic Delaunay triangulations and periodic hyperbolic Delaunay triangulations are implemented as two separate packages of CGAL and are publicly available on GitHub, on the `cgal-public-dev` repository of CGAL. Both packages are currently under revision for integration into CGAL. After they have been merged into the library, they will be removed from `cgal-public-dev`. For this reason, information about the current status of the submission, as well as links to the actual code, are provided at the following link:

`https://imiordanov.github.io/code/`

IV.1 A short introduction to CGAL

The CGAL project has started in 1996, aiming to provide robust implementations for geometric algorithms. The collection of these implementations is a C++ library bearing the same name. CGAL incorporates data structures and algorithms for triangulations,

meshes, arrangements, geometric optimization, shape reconstruction, classification, and many other. The library is open source and is distributed freely, some of its packages under the GNU General Public License (GPL) v3+, and others under the GNU Lesser General Public License (LGPL) v3+. Commercial licenses are provided by the company GeometryFactory.[12]

IV.1.1 Predicates and Exact Geometric Computation

One of the advantages of CGAL is that all the implementations it includes are *robust*. The need for robustness in numerical computations is manifested in all aspects of science and technology, but is really accentuated in geometric computations. From the CGAL point of view, a program is robust if it does not crash independently on the input, is capable of handling degenerate input (for instance, computing the Delaunay triangulation of points that lie on a grid), and its output is always correct. In CGAL, robustness is achieved via the so-called *Exact Geometric Computation* (EGC) paradigm, pioneered by Chee Yap [YD95].[13] In the example of a set of points on a grid, a computation of their Delaunay triangulation may be wrong or inconsistent, or the program may even crash, if issues due to inexact arithmetic are not taken into account: real numbers cannot be represented in a computer exactly, but are instead represented with finite accuracy, so computations with such numbers are exact up to finite accuracy as well.

Specifically for the construction of triangulations, the adoption of EGC is reflected in the fact that geometric algorithms are formulated in terms of predicates, which were discussed in Section III.4 in the context of computing Delaunay triangulations of the Bolza surface. Predicates are simple geometric questions whose answer usually amounts to determining the sign of a polynomial in the coordinates of input points. If this sign can be determined with certainty, then the result of the algorithm is always correct. It becomes then a separate issue to guarantee that predicates are evaluated exactly, for example by using special number types that allow to represent numbers with arbitrary accuracy.

It should be noted that EGC does not imply exact arithmetic. CGAL tracks error bounds and resorts to extended precision only when it is really needed, because the end goal is not to compute numbers exactly. Instead, the goal of a geometric algorithm is to compute a geometric structure (in our case, a triangulation), and EGC guarantees that the computed structure is correct.

Each algorithm for the computation of Delaunay triangulations uses a different set of predicates. The incremental algorithm by Bowyer [Bow81, Wat81] only needs two predicates, ORIENTATION and INCIRCLE. The exact evaluation of these two predicates, coupled with well-designed implementations that handle degenerate cases properly, guar-

[12] https://geometryfactory.com/

[13] Details about the Exact Geometric Computation paradigm and robustness issues can be found on the CGAL website: https://www.cgal.org/exact.html

antee that the construction of Delaunay triangulations is robust.

IV.1.2 Techniques adopted by CGAL

In this section we discuss several standards that are followed by CGAL, in order to gain a better insight into the structure of the library, and also to familiarize the reader with the ideas that we discuss in the next sections.

Object-oriented programming

CGAL follows the Object-Oriented Programming (OOP) paradigm, which is based on the notion of *objects*. A programming object is an abstract description of a real-world object. Each object is described in a *class*, which specifies the *properties*, or *attributes*, of the object, as well as its *methods*. The methods of an object describe actions on its properties. Properties and methods can be accessed at multiple levels, the most common being `public` (accessible by any other object) and `private` (accessible only by objects of the same class). It is usual practice to declare the properties of an object privately, and provide public access methods. An example of an object that we will encounter is `Vertex`, which represents a vertex of a triangulation. The properties of a vertex are the geometric point (of type `Point_2`) that it stores, and the triangulation face to which it points (of type `Face`). Both of these properties are `private` to the `Vertex` class, so they are not directly accessible. To retrieve or modify the point stored in a vertex or the face to which it points, the class provides appropriate public methods.

One of the most important features of OOP is the notion of *inheritance*. Inheritance allows objects to be defined hierarchically, by deriving properties and methods from a more abstract *base class*. For example, a `Delaunay_triangulation_2` in CGAL inherits from a `Triangulation_2`, which means that it *is* a `Triangulation_2`. This "is-a" relationship indicates inheritance. In fact, a `Delaunay_triangulation_2` has all the properties and methods of a `Triangulation_2`, but *overloads* some methods or adds new ones. A very good example of an overloaded method is the insertion of new points, which in `Delaunay_triangulation_2` makes sure that the triangulation remains a Delaunay triangulation.

In this chapter, we use common `C++` notation to indicate inheritance. The snippet of code below shows the declaration of the class `Delaunay_triangulation_2` in its basic form.

```
class Delaunay_triangulation_2: public Triangulation_2
  {
    // code here
  };
```

The keyword `public` indicates that `Delaunay_triangulation_2` inherits all the public properties and methods of `Triangulation_2`. In programming terms, public inheritance

indicates the "is-a" relationship that we mentioned before. It is possible to inherit privately as well.

Generic programming and concepts

CGAL adopts a powerful abstraction supported by C++, called *generic programming*. The philosophy of generic programming is that, in order to execute a specific operation on an object, one does not need to know the specific type of object itself, but rather the operations that the object supports. For instance, suppose that we need to sort a set of given objects of the same type. We do not need to know the type of the objects, we just need to know that they are all of the same type and can be compared, i.e., we must have a way to know that a given object is "larger" than another object. Therefore, the class of these objects must define a comparison method. The philosophy of generic programming is that the same sorting function can be applied to *any* type of object that provides such a comparison method.

Such abstract requirements on objects are encapsulated in *concepts*. A concept is a set of requirements (either properties, methods, or both) on some type used either as input to a function or to define a different type of object. It is of practical advantage to keep the number of requirements to a minimum for two reasons. On one hand, the hierarchy and separation between different concepts becomes clearer and more natural. On the other hand, specifying only requirements that are absolutely necessary in a single concept helps to avoid superfluous coding. A class that fulfills all the requirements specified by a concept is a *model* of that concept. Note that a single class can be model of multiple concepts.

Generic programming in CGAL is implemented via *template arguments* given as input to the object or function that is being declared. A brief example of using concepts is given in the code snipped below, in which we declare a generic sorting function.

```
template <typename GenericType>
void bubble_sort(std::vector<GenericType>& elements)
  {
    for (int i = 0; i < elements.size() - 1; i++) {
      for (int j = i+1; j < elements.size(); j++) {
        if ( elements[i].less_than(elements[j]) ) {
          std::swap(elements[i], elements[j]);
        }
      }
    }
  }
```

The keyword `typename` indicates that the identifier that follows is the name of a type. The notation `std::vector` indicates that `vector` is a type defined in the *namespace* `std`

(the C++ standard library).[14] From the declaration of the function `bubble_sort`, we infer that `GenericType` is a template parameter that requires a method named `less_than` that accepts one argument of the same type `GenericType`. Also, the method `less_than` returns a Boolean value indicating whether the value of its argument is "less" than the value of the object on which it is called. Any vector with elements whose type is a model of the concept `GenericType` is valid input for the function `bubble_sort`, so the function can be used to sort numbers, points, segments, and any other objects for which comparison is defined.

IV.2 Triangulations in CGAL

Triangulations find applications in numerous fields of science and technology, and for this reason they have been present in CGAL from the very first release of the library. We begin this section by discussing the triangulation packages that currently exist in CGAL, while we give also a timeline of their integration. We will then see how a triangulation is declared, which will allow us to explore the hierarchy of objects that are involved in the declaration. In the rest of the section we discuss each object and give typical examples of use.

IV.2.1 Historical notes

The first version of the CGAL library was released in 1997, and it included an implementation of triangulations in 2 dimensions by Yvinec [Yvi97]. At the beginning of 2000, Teillaud introduced triangulations in 3 dimensions to the library, with Pion and Jamin contributing in several ways in successive versions [JPT14]. Generalized triangulations in $d \geq 2$ dimensions were first added by Hert and Seel at the end of 2001, but were replaced in 2015 by a package developed initially by Hornus and later integrated by Devillers and Jamin [DHJ15].

Caroli worked on Euclidean periodic triangulations[15] in 3 dimensions for his PhD thesis with Teillaud, and the code he produced was integrated into CGAL in 2009 [CT09]. Euclidean periodic triangulations in 2 dimensions followed in 2013 by Kruithof, drawing inspiration from the 3D implementation [Kru13]. Both packages for Euclidean periodic triangulations allow to work either with covering spaces or with dummy points for point sets that do not satisfy the validity condition (II.31).

IV.2.2 Declaration of a triangulation

We give now an overview of two-dimensional triangulations in CGAL. Similarly to other data structures in CGAL, triangulations maintain a clear separation between geometry

[14] The same notation is used to derive types from objects, as we will see in subsequent examples.
[15] Recall that Euclidean periodic triangulations are triangulations of the flat torus.

and topology. This separation is reflected in the fact that all triangulation classes take two template parameters, one *geometric traits*, and one *triangulation data structure* (or TDS for short). In Snippet IV.1 we show the definition of the class `Triangulation_2`.

```
template< class GT,
          class TDS = Trangulation_data_structure_2<> >
Triangulation_2<GT, TDS>
  {
    // properties, methods, ...
  };
```

Snippet IV.1: Declaration of the class `Triangulation_2` in CGAL

We will discuss concepts and models for geometric traits in Section IV.2.3 and TDS in Section IV.2.5. The traits class provides geometric objects and operations on them, while the TDS manages the combinatorial structure of the triangulation. The triangulation class itself provides high-level operations, such as point location and insertion, or the removal of existing vertices. For each operation it needs to take into account both geometric and combinatorial information.

In Snippet IV.2 we give an example of how a typical `Triangulation_2` is defined.

`Triangulation_data_structure_2` takes two template parameters, one *vertex base* class (by default `Triangulation_vertex_base_2`) and one *face base* class (by default `Triangulation_face_base_2`) that represent the vertices and faces of the triangulation, respectively. We will discuss in more detail these classes, as well as the ones with `_ds_` in their names, in Section IV.2.6.

For the GT template parameter, in most cases one of the *kernels* provided by CGAL can be used directly: a kernel is usually a model of a geometric traits concept in CGAL. We discuss kernels in more detail in Section IV.2.4. Some kernels in CGAL take one template parameter, a *field number type*, or FT for short, that specifies the type of numbers used to represent properties of geometric objects (for example, the coordinates of points). In the example shown in Snippet IV.2, we use `Cartesian` as `Kernel`, with `CORE::Expr` as FT.

```
typedef Cartesian<CORE::Expr>                                  Kernel;
typedef Kernel                                              Geom_traits;

typedef Triangulation_ds_vertex_base_2<>                       DS_vb;
typedef Triangulation_vertex_base_2<Geom_traits,DS_vb>  Vertex_base;

typedef Triangulation_ds_face_base_2<>                         DS_fb;
typedef Triangulation_face_base_2<Geom_traits, DS_fb>     Face_base;

typedef Triangulation_data_structure_2<Vertex_base, Face_base> TDS;
typedef Triangulation_2<Geom_traits, TDS>                  Triangulation;
```

```
Triangulation tr;
```

Snippet IV.2: Definition of a triangulation in CGAL

The example in Snippet IV.2 shows explicit declarations of the default objects used by CGAL to define a triangulation. A triangulation of the same type can be declared in a much simpler way by omitting these explicit declarations, as shown in Snippet IV.3.

```
typedef Cartesian<CORE::Expr>                    Kernel;
typedef Triangulation_2<Kernel>                  Triangulation;

Triangulation tr;
```

Snippet IV.3: Definition of a triangulation in CGAL *using the default model for* TDS

IV.2.3 Geometric traits

Geometric traits, or simply traits, are of major importance in CGAL. Each package of the library has at least one traits concept, and each traits concept has at least one default model. Traits concepts describe a set of requirements concerning geometric objects and operations. For triangulations, the required objects include points, segments, lines, etc. Operations are separated in two categories, constructions and predicates. We have already discussed predicates, while an example of a construction is the circumcenter of three points. As we mentioned in Section IV.2.2, it is not unusual for a CGAL kernel to be model of a geometric traits concept.

IV.2.4 Kernels

Geometric kernels in CGAL provide basic geometric objects (e.g., points, segments, lines, circles) and define basic operations on them (both constructions and predicates). They can be used directly as geometric traits classes in the declarations of triangulations, as we saw in Snippet IV.3.

Different kernels may provide different representations for basic objects. For instance, the `Cartesian` and the `Homogeneous` kernels define the same objects and the same operations on them, but in one objects are represented using Cartesian coordinates, while in the other using homogeneous coordinates. Both `Cartesian` and `Homogeneous` are models of the concept `Kernel`.

Another difference that kernels may have lies in the type of operations they provide. For instance, the concept `CircularKernel` refines `Kernel` by adding more operations on circles and circular arcs in the Euclidean plane, so its model `Circular_kernel_2` provides a much wider range of operations on such objects than `Cartesian`.

IV.2.5 Triangulation data structure

Recall from Snippet IV.1 that `Triangulation_data_structure_2` is given as default template parameter for `TDS` in the declaration of `Triangulation_2`. In fact, the class `Triangulation_data_structure_2` is the default model offered by CGAL for the `TDS` concept. It takes two template parameters, a vertex base class and a face base class. `Triangulation_data_structure_2` describes the combinatorial structure of a two-dimensional triangulation in CGAL via its faces and vertices. It defines the types of objects that represent combinatorial elements of the triangulation (vertices, edges, and faces) and operations on them (setting and retrieving adjacency and incidence relations). It provides *iterators* over the faces, edges, and vertices of the triangulation, and maintains the combinatorial correctness of the triangulation during each modification.

A vertex stores a pointer to one of its incident faces. A face stores three pointers to its incident vertices, and also three pointers to its neighboring faces. Faces index their incident vertices with indices $0, 1, 2$. The i-th neighbor of a face is the one opposite to the i-th vertex of the face. Edges are not explicitly stored, but are instead given as a pair of a face and the index of the vertex opposite to the edge. This implies that an edge is not uniquely specified, as it is shared by two faces. See Figure IV.4.

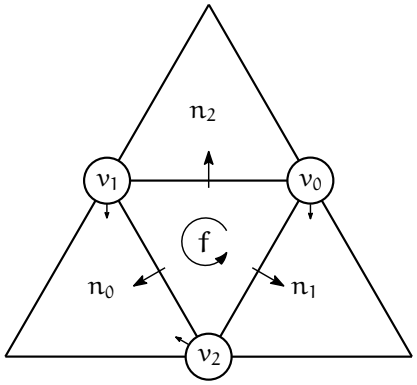

Figure IV.4: Illustration of the CGAL *triangulation data structure. Each vertex* $v_i, i = 0, 1, 2$ *gives access to one of its incident faces. Each face gives access to its incident vertices and to its adjacent faces* $n_i, i = 0, 1, 2$.

IV.2.6 Faces and vertices

The separation between geometry and combinatorics is reflected also in the concepts and models for faces and vertices of a triangulation in CGAL. Recall Snippet IV.2 for an illustration of the dependencies that we discuss in this section.

Combinatorial operations on vertices and faces are provided by the classes `Triangulation_ds_vertex_base_2` and `Triangulation_ds_face_base_2` respectively. In the names of these classes, `_ds_` stands for *data structure*. For a vertex, the opera-

tions provided are setting and retrieving a face of the triangulation to which the vertex is incident. For a face, the operations are setting and retrieving the vertices incident to the face, and also setting and retrieving the faces adjacent to it. Both classes take a triangulation data structure as template parameter, for which `Triangulation_data_structure_2` is used by default. See Figure IV.5.

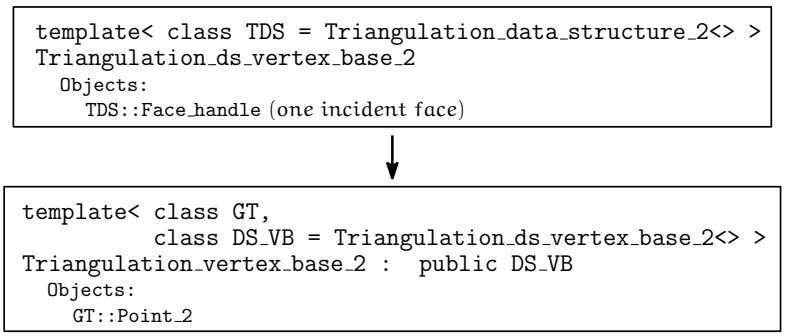

Figure IV.5: Vertex inheritance diagram.

Geometric operations for vertices and faces are provided respectively by the classes `Triangulation_vertex_base_2` (which inherits from `Triangulation_ds_vertex_base_2`) and `Triangulation_face_base_2` (which inherits from `Triangulation_ds_face_base_2`). For a vertex, the geometric operations enable the storage and retrieval of a point. For a face, the class `Triangulation_face_base_2` adds no functionality: it is used to preserve the symmetry with the `Triangulation_vertex_base_2` class. Both classes take two template parameters: one geometric traits class and one `_ds_` base. The traits class has no default model. The `_ds_` base for `Triangulation_vertex_base_2` is by default `Triangulation_ds_vertex_base_2`, while for `Triangulation_face_base_2` the default `_ds_` base is `Triangulation_ds_face_base_2`.

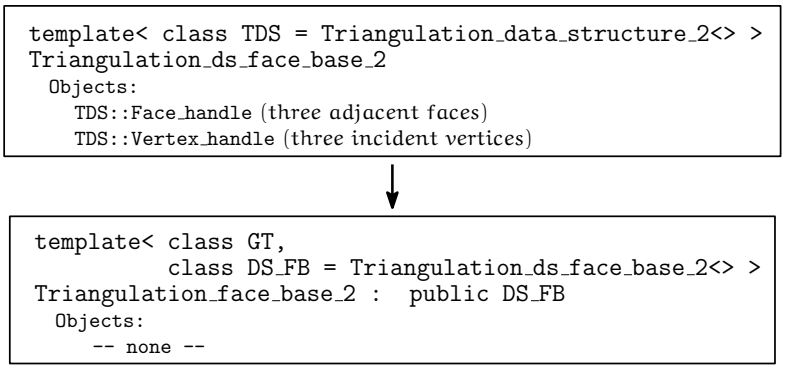

Figure IV.6: Face inheritance diagram.

IV.3 Periodic hyperbolic triangulations

In this section we discuss the objects that we implement and integrate into CGAL for the computation of triangulations of the Bolza surface. The names of the classes that we introduce in most cases begin with the prefix `Periodic_4` to indicate that a triangulation of the Bolza surface, seen as a periodic triangulation of $\mathbb{H}^2$, is periodic in the four directions defined by the generators of Γ_2.

IV.3.1 Triangulation objects

A Delaunay triangulation of the Bolza surface is represented by an object named `Periodic_4_hyperbolic_Delaunay_triangulation_2`. In accordance with established CGAL practices, we introduce also a lower-level class named `Periodic_4_hyperbolic_triangulation_2`, which serves as base class for the Delaunay triangulation, and possibly other types of triangulations that may be added in the future (for example, constrained Delaunay triangulations[16]). We discuss here details for both objects and the relationship between them.

Similarly to other triangulation classes in CGAL, the class `Periodic_4_hyperbolic_triangulation_2` takes two template parameters, one geometric traits and one `TDS`. See Snippet IV.7.

```
template<
  class GT,
  class TDS = Triangulation_data_structure_2<
    Periodic_4_hyperbolic_triangulation_vertex_base_2< GT >,
    Periodic_4_hyperbolic_triangulation_face_base_2< GT >
  > >
Periodic_4_hyperbolic_triangulation_2<GT, TDS>
  {
    // properties, methods, ...
  };
```

Snippet IV.7: Declaration of the class `Periodic_4_hyperbolic_triangulation_2` in CGAL

The geometric traits must be provided, as there is no default for the template parameter. We provide a suitable model for this template parameter, which we discuss in Section IV.3.2. The vertex and face base classes used in the definition of the default `TDS` model are discussed in Section IV.3.4. The class `Periodic_4_hyperbolic_triangulation_2` does not support insertion of points or removal of vertices. It defines basic geometric and combinatorial objects (points, vertices, faces, hyperbolic translations), which are inferred from the traits and TDS classes. From

[16] A *constrained Delaunay triangulation* is a Delaunay triangulation that contains some predetermined edges, given as input.

the TDS class it also exposes iterators over the vertices, edges, and faces of the triangulation. From the traits class, it exposes construction and predicate objects (orientation and in-circle tests, application of a hyperbolic translation in $\mathcal{N}_2$ to a point, construction of dual objects). In this class we also implement location functions, both Euclidean and hyperbolic (recall Sections III.2.2 and III.3.1).

The class `Periodic_4_hyperbolic_Delaunay_triangulation_2` inherits from `Periodic_4_hyperbolic_triangulation_2` and also takes two template parameters, one geometric traits and one `TDS`. The default model for `TDS` is as for `Periodic_4_hyperbolic_triangulation_2`, while the default geometric traits is `Periodic_4_hyperbolic_Delaunay_triangulation_traits_2`. In the class `Periodic_4_hyperbolic_Delaunay_triangulation_2` we implement the following operations:

- initialization with the set of rational dummy points (recall Section III.2.1);
- insertion of new points in the triangulation;
- removal of unnecessary dummy points (either automatic or via explicit function calls);
- removal of existing vertices;
- construction of Voronoi vertices and edges.

Figure IV.8 shows a schematic representation of the inheritance of the class `Periodic_4_hyperbolic_Delaunay_triangulation_2` from `Periodic_4_hyperbolic_triangulation_2`. All the objects, predicates, and operations defined in `Periodic_4_hyperbolic_triangulation_2` are available in `Periodic_4_hyperbolic_Delaunay_triangulation_2`.

IV.3.2 Periodic hyperbolic geometric traits

We start with some context relevant to the geometric traits class that we introduce.

Work on Delaunay triangulations in the hyperbolic plane $\mathbb{H}^2$ and their implementation in CGAL started with the PhD thesis of Bogdanov [BDT14]. Following the established CGAL practice, Bogdanov and Teillaud encapsulated the requirements for a suitable geometric traits class in the concept `HyperbolicDelaunayTriangulationTraits_2`, for which they also provided a model. Their traits class, named `Hyperbolic_Delaunay_triangulation_CK_traits_2`, takes a circular kernel as template parameter for which the default model is

```
Circular_kernel_2<
  Exact_predicates_inexact_constructions_kernel,
  Algebraic_kernel_for_circles_2_2<
    Exact_predicates_inexact_constructions_kernel::RT > >.
```

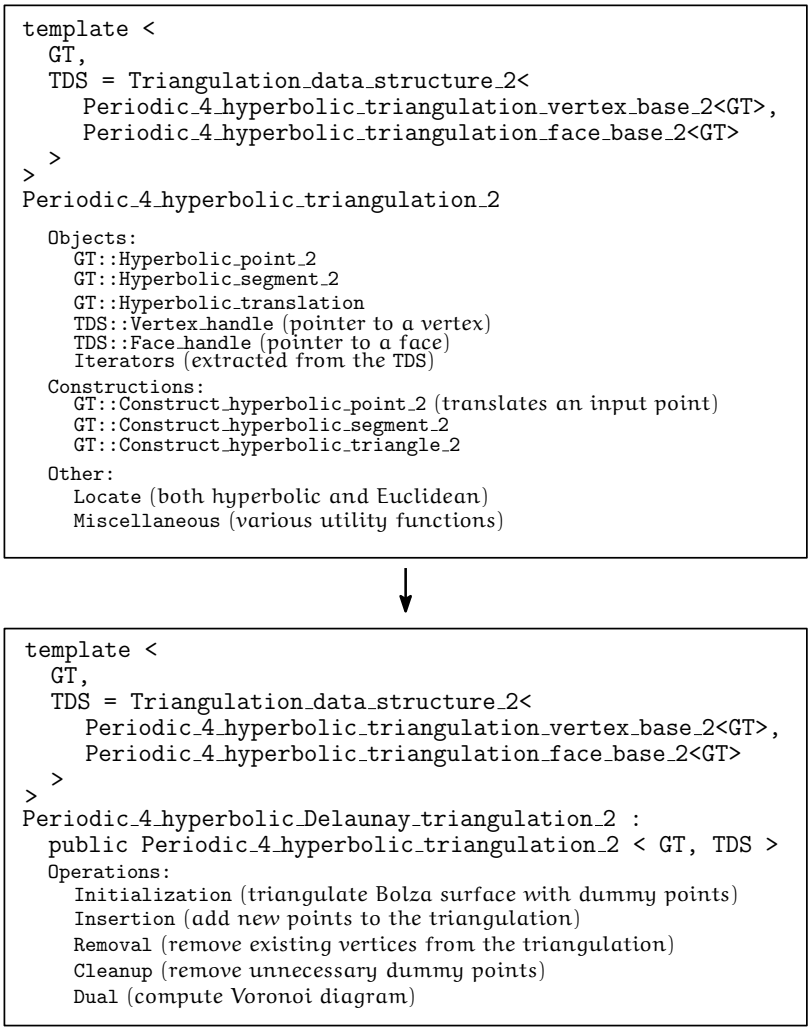

Figure IV.8: Inheritance diagram for periodic hyperbolic triangulation objects.

This traits class provides exact constructions and predicates for input points with rational coordinates.

For the construction of periodic Delaunay triangulations of $\mathbb{H}^2$, we *refine* the concept HyperbolicDelaunayTriangulationTraits_2 into the concept Periodic_4HyperbolicDelaunayTriangulationTraits_2, which adds supplementary requirements for hyperbolic translations. In terms of classes, our model of Periodic_4HyperbolicDelaunayTriangulationTraits_2 should inherit from a model of HyperbolicDelaunayTriangulationTraits_2. However, we need exact predicates for points with algebraic coordinates (recall Section III.4), so Hyperbolic_Delaunay_triangulation_CK_traits_2 is not a suitable base class. Therefore, we implement a second

model of the concept `HyperbolicDelaunayTriangulationTraits_2`, named `Hyperbolic_Delaunay_triangulation_traits_2`, and we use it as base class for `Periodic_4_hyperbolic_Delaunay_triangulation_traits_2`, which we provide as a model of the concept `Periodic_4HyperbolicDelaunayTriangulationTraits_2`. `Hyperbolic_Delaunay_triangulation_traits_2` takes a kernel as template parameter that is by default `Cartesian<CORE::Expr>`.

The two traits classes `Hyperbolic_Delaunay_triangulation_CK_traits_2` and `Hyperbolic_Delaunay_triangulation_traits_2` provide the basic objects (points and hyperbolic segments) and operations (constructions and predicates) necessary for the computation of Delaunay triangulations in the hyperbolic plane. To represent a hyperbolic segment, both classes use an object called `boost::variant` that can be either a Euclidean segment or a Euclidean circular arc. The supported constructions are:

- hyperbolic segments,
- hyperbolic bisectors,
- intersections of hyperbolic segments,
- and hyperbolic circumcenters.

Both traits classes expose the Euclidean InCircle and Orientation predicates from their respective kernels. These predicates are used for the construction of Delaunay triangulations in $\mathbb{H}^2$: the InCircle predicate is used to identify faces in conflict with an input point, and the Orientation predicate to perform a visibility walk in search of an initial face in conflict with an input point. We added subsequently the SideOfOrientedHyperbolicSegment predicate (which is expressed in terms of the Orientation and InCircle predicates) to the concept `HyperbolicDelaunayTriangulationTraits_2` and implemented it in both of its models. This predicate enables the implementation of point location functions (recall Section III.4.1).

As we mentioned earlier, we provide the model `Periodic_4_hyperbolic_Delaunay_triangulation_traits_2` for the concept `Periodic_4HyperbolicDelaunayTriangulationTraits_2`. Our traits class inherits from `Hyperbolic_Delaunay_triangulation_traits_2` and adds the following components:

- An object to represent hyperbolic translations of the set $\mathcal{N}_2$;
- Definitions of all operations (constructions and predicates) that involve hyperbolic translations;
- Approximate computation of the circumdiameter of a face (needed to verify if faces of the Delaunay triangulation of $\mathbb{M}_2$ violate the validity condition (II.36) during the removal of vertices);
- The SideOfOriginalOctagon predicate.

The class `Periodic_4_hyperbolic_Delaunay_triangulation_traits_2` takes a ker-

nel as template parameter, which defaults to `Cartesian<CORE::Expr>`. The inheritance of the periodic hyperbolic traits class from the hyperbolic traits class is shown in Figure IV.9.

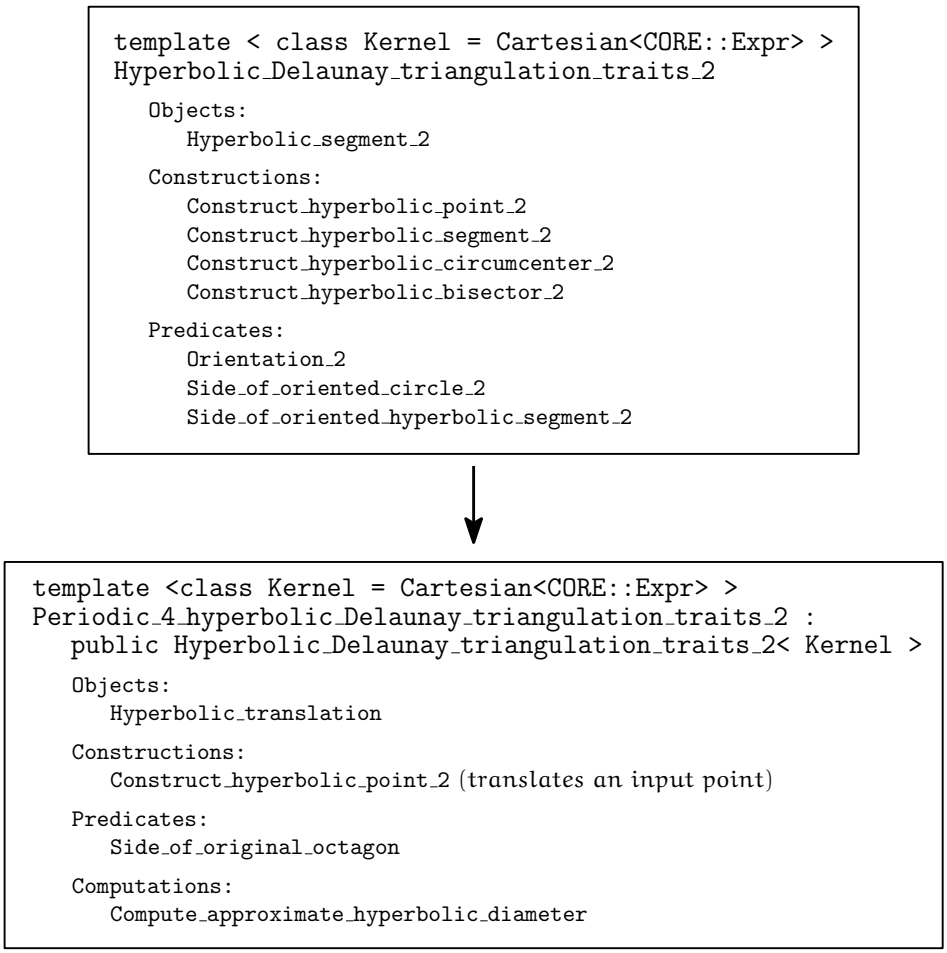

Figure IV.9: Inheritance diagram for the periodic hyperbolic traits object.

IV.3.3 Hyperbolic translations

In Section III.1.2, we discussed how hyperbolic translations in the set $\mathcal{N}_2$ can be seen as words on the alphabet $\mathcal{A}$, and how we can use Dehn's algorithm to obtain a unique representation for each such translation. In this section we discuss the actual representation of these words in our code. We also discuss how we retrieve the coefficients α and β for the matrix of each translation in $\mathcal{N}_2$ (recall expression (II.14)). Finally, we introduce an object that represents a hyperbolic translation in $\mathcal{N}_2$ both as a word and as a matrix, giving access to both word simplification and to the coefficients of its matrix.

Words. We discuss first the representation of words on the alphabet $\mathcal{A}$. Recall from (II.21) that the relation of Γ_2 can be written as $\mathcal{R}_2 = T_0T_5T_2T_7T_4T_1T_6T_3$; note that for any two consecutive translations T_i and T_j in this sequence, $j = (i+5) \bmod 8$. We represent each generator of Γ_2 via its integer index from 0 to 7. Recall that, with this notation, the inverse of the generator T_k of Γ_2 is the generator T_{k+4}, where indices are considered modulo 8. So, a word on the alphabet $\mathcal{A}$ is encoded as a sequence of integers in $\mathbb{Z}_8$, and the relation $\mathcal{R}_2$ is encoded as the sequence 05274163.

With this encoding, we describe the simplification of words on $\mathcal{A}$ with Dehn's algorithm. Recall from Section III.1.2 that the simplification is done in two steps, successive free reduction and simplification of subsequences of $\mathcal{R}_2^\infty$ of $(\mathcal{R}_2^\infty)^{-1}$, until no further simplification is possible. The free reduction step consists in progressively identifying and removing pairs of consecutive numbers in the input word with values k and $k \pm 4$ modulo 8. Then, the identification of sequences of letters in $\mathcal{A}$ that are subsequences of $\mathcal{R}_2^\infty$ or $(\mathcal{R}_2^\infty)^{-1}$ is based on the fact that a sequence of letters $w_n = (T_{k_j})_{j=0,1,\dots,n}$, $T_{k_j} \in \mathcal{A}$, is a sub-word of $\mathcal{R}_2^\infty$ of length n if, for every j from 0 to $n-1$, $k_{j+1} = (k_j + 5) \bmod 8$. Similarly, (T_{k_j}) is a sub-word of $(\mathcal{R}_2^\infty)^{-1}$ of length n if for every j from 0 to $n-1$, $k_{j+1} = (k_j - 5) \bmod 8$. If a subsequence w_n of $\mathcal{R}_2^\infty$ or $(\mathcal{R}_2^\infty)^{-1}$ is identified, we take cases for its length n:

- if $n = 8$, then w_n is eliminated;
- if $4 < n < 8$, then w_n is replaced by its reduced version t^{-1}, which is such that $w_n t = \mathrm{Id}$ and is necessarily shorter than w_n;
- if $n = 4$ and also w_n is a subsequence of $(\mathcal{R}_2^\infty)^{-1}$, then w_n is replaced by its inverse word w_n^{-1};
- else, w_n is left unaltered.

As input, our implementation of Dehn's algorithm takes a sequence of integers in $\mathbb{Z}_8$ that is the concatenation of two or three words of $\mathcal{N}_2$ (recall Sections III.2.2 and III.2.3). The sequence is then simplified as described in the previous paragraph until no further simplification is possible. The periodicity of the Bolza surface is such that the result of simplifying a combination of words in $\mathcal{N}_2$ is always a word in $\mathcal{N}_2$, so the output is a sequence of integers in $\mathbb{Z}_8$ of length at most four. The simplification of words with this slightly modified version of Dehn's algorithm gives us a *unique* representative for each translation.

To store and handle words corresponding to elements of $\mathcal{N}_2$, we introduce a class named `Hyperbolic_octagon_translation_word` that stores four integers in $\mathbb{Z}_8$. The class allocates 3 bits to each stored integer, which physically limits the possible values for each stored integer to $\mathbb{Z}_8$.[17] The value of each integer is manually initialized to 0. This initialization causes a problem when we need to differentiate between the translation T_0 and the identity translation Id. To deal with this problem,

[17] The bit size of variables can be manually set in C++ with the use of *bit fields*.

`Hyperbolic_octagon_translation_word` stores also four 1-bit boolean flags, one for each integer, to indicate whether the integer is "active" or not. For the identity translation, all integers are set to 0, and all flags are inactive. For the translation T_0, all integers are 0 and only the first flag is active. This representation allows us to store each word of $\mathcal{N}_2$ by using only 2 bytes of memory. The operations supported by the class `Hyperbolic_octagon_translation_word` are:

- inversion,
- concatenation (whose result is simplified with Dehn's algorithm),
- and comparison.

The comparison of two words in $\mathcal{N}_2$ is done according to the ordering of the elements of $\mathcal{N}_2$, introduced in Section III.1.1.

Matrices. To retrieve the matrix coefficients for each translation in $\mathcal{N}_2$, we introduce a class called `Hyperbolic_octagon_translation_matrix`. This class stores the two coefficients α and β of the matrix of a hyperbolic translation (recall (II.14)), and defines the multiplication operation for two matrices.

Translations. Our goal is to have a representation for hyperbolic translations that enables both simplification with Dehn's algorithm and retrieval of the matrix coefficient of the corresponding translation (which is needed to apply the translation to an input point). We introduce a class called `Hyperbolic_octagon_translation` that provides both functionalities. Each `Hyperbolic_octagon_translation` object contains one object of type `Hyperbolic_octagon_translation_word`, and links to a corresponding object of type `Hyperbolic_octagon_translation_matrix` contained in a map between words and matrices. This map is private to the class `Hyperbolic_octagon_translation` and is calculated once, mapping each word in $\mathcal{N}_2$ to its corresponding matrix. This initialization enables us to avoid computing matrix multiplications in the sequel: the multiplication of two `Hyperbolic_octagon_translation` objects amounts to concatenating the respective `Hyperbolic_octagon_translation_word` objects, which results in updating the link to the corresponding `Hyperbolic_octagon_translation_matrix` object in the map.

```
template< class FT = CORE::Expr >
class Hyperbolic_octagon_translation
  {
  private:
    Hyperbolic_octagon_translation<FT>                Self;
    typedef unsigned short int                        Int;
    typedef Hyperbolic_octagon_translation_word<Int>  Word;
    typedef Hyperbolic_octagon_translation_matrix<FT> Matrix;
```

```
    Word                                _w;
    Matrix                              _m;
    static std::map<Word, Matrix>       _translations_map;

    ...

  public:

    ...

    // The multiplication operator concatenates the words that
    // correspond to both objects. Note that the result of the
    // concatenation is simplified internally.
    Self operator*(const Self& rh) const {
      return Self(this->_w * rh._w);
    }

    Matrix matrix() const {
      return _translations_map[this->w];
    }
  };
```

Snippet IV.10: Definition of hyperbolic translations

The class `Hyperbolic_octagon_translation` takes a field number type `FT` as template parameter that determines the number type of the matrix coefficients of the translation, as can be seen from Snippet IV.10. For the whole implementation to work correctly, it is necessary that `FT` provides exact operations with algebraic numbers, specifically with nested square roots. By default, `FT` is `CORE::Expr`. The declaration of hyperbolic translation objects is done in the traits class `Periodic_4_hyperbolic_Delaunay_triangulation_traits_2` and is shown in Snippet IV.11.

```
  template< class Kernel = CGAL::Cartesian<CORE::Expr> >
  class Periodic_4_hyperbolic_Delaunay_triangulation_traits_2
    : public Hyperbolic_Delaunay_triangulation_traits_2<Kernel>
    {
      typedef Hyperbolic_octagon_translation<Kernel::FT>
                                          Hyperbolic_translation;
      ...
    };
```

Snippet IV.11: Declaration of hyperbolic translation objects in the periodic hyperbolic traits class

IV.3.4 Periodic hyperbolic vertices and faces

We discussed in Section IV.2.6 the hierarchy of the concept and classes used to represent vertices and faces in a general triangulation of CGAL. Here we describe the classes we introduce to represent vertices and faces of a Delaunay triangulation of the Bolza surface.

We introduce a class named `Periodic_4_hyperbolic_triangulation_vertex_base_2`. This class inherits from `Triangulation_vertex_base_2`, and adds an interface for storing and retrieving hyperbolic translations. This interface is essential for the insertion and removal procedures, which we discussed in Sections III.2.3 and III.3.2. The class `Periodic_4_hyperbolic_triangulation_vertex_base_2` takes two template parameters, a geometric traits and a vertex base. There is no default geometric traits, while the vertex base is by default `Triangulation_vertex_base_2`. The hierarchy between vertex objects is shown schematically in Figure IV.12.

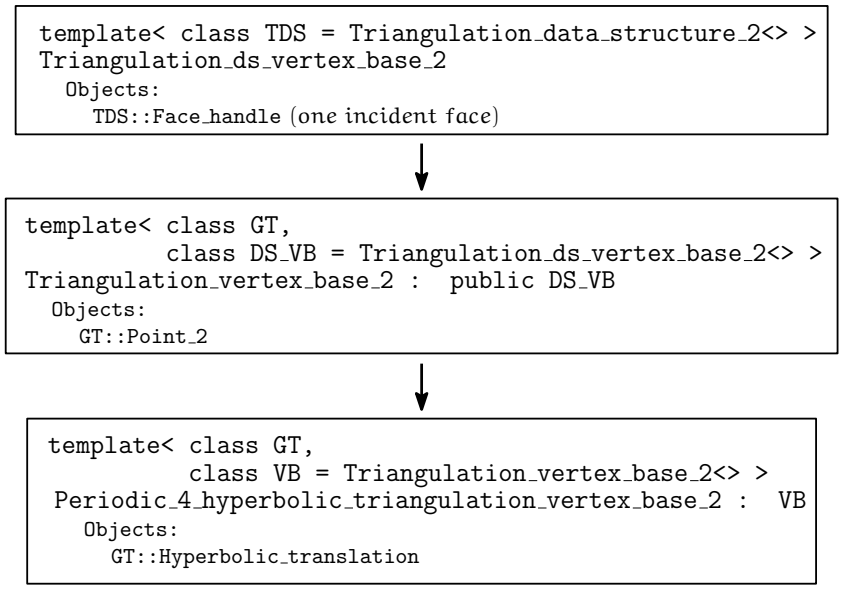

Figure IV.12: Inheritance diagram for vertex objects.

To represent a face of the periodic hyperbolic triangulation, we similarly introduce a new class called `Periodic_4_hyperbolic_triangulation_face_base_2`. This object inherits from `Triangulation_face_base_2` and adds an interface for the storage and retrieval of three hyperbolic translations in the face, one for each vertex. It provides functions to set, modify, and retrieve the hyperbolic translation associated with each vertex. The class `Periodic_4_hyperbolic_triangulation_face_base_2` takes two template parameters, a geometric traits and a face base. There is no default geometric traits, while the default face base is `Triangulation_face_base_2`. The relationship between the `Periodic_4_hyperbolic_triangulation_face_base_2` class and existing face classes is shown in Figure IV.13.

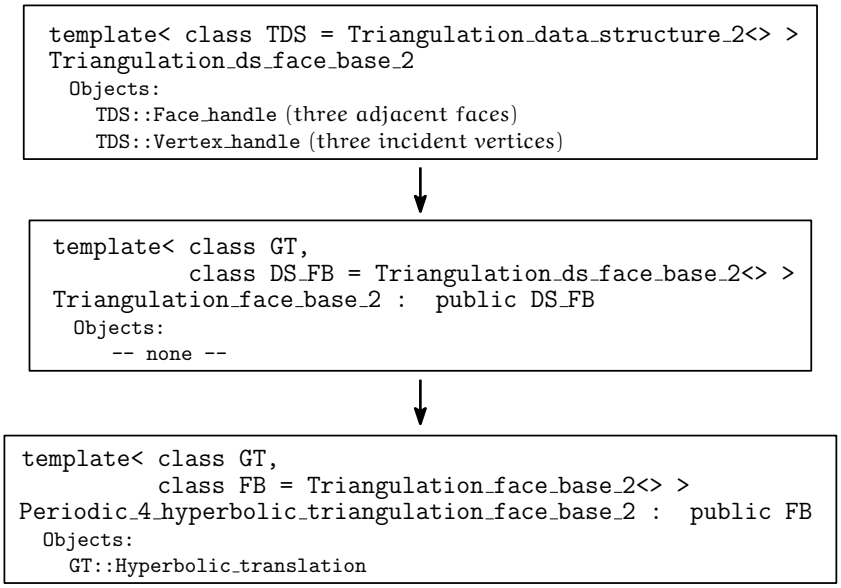

Figure IV.13: Inheritance diagram for face objects.

IV.4 Application demo

Most packages in CGAL include a *demo*, which is a small stand-alone application with a Graphical User Interface (GUI) that allows the user to directly visualize the functionalities offered by the package. Triangulation packages in CGAL include demos that allow the computation of triangulations either from input files that contain the coordinates of a set of points, or by manually clicking on an area on the screen, which causes the insertion of new points in the triangulation.

In our implementation, we include a demo for Delaunay triangulations of the Bolza surface. The execution of the demo application creates a window showing the Poincaré disk with the closed hyperbolic octagon D_2 and the triangulation of $\mathbb{M}_2$ defined by the set of rational dummy points $\mathcal{Q}'_2$, introduced in Section III.2.1. See Figure IV.14. The user is immediately able to insert new points by clicking inside the octagon. Each click produces a point p, and if p lies inside the octagon, then it is inserted in the triangulation. If p is not inside the octagon, then the click is simply ignored. After the insertion of each new point, the unnecessary dummy points (if any) are removed from the triangulation.

The GUI of the demo (shown in Figure IV.14) allows the user to execute a variety of other actions:

- **Clear triangulation:** re-initialize the triangulation with the set of dummy points;
- **Open input file:** insert points from a file into the triangulation;
- **Save to file:** export the triangulation to a text file;

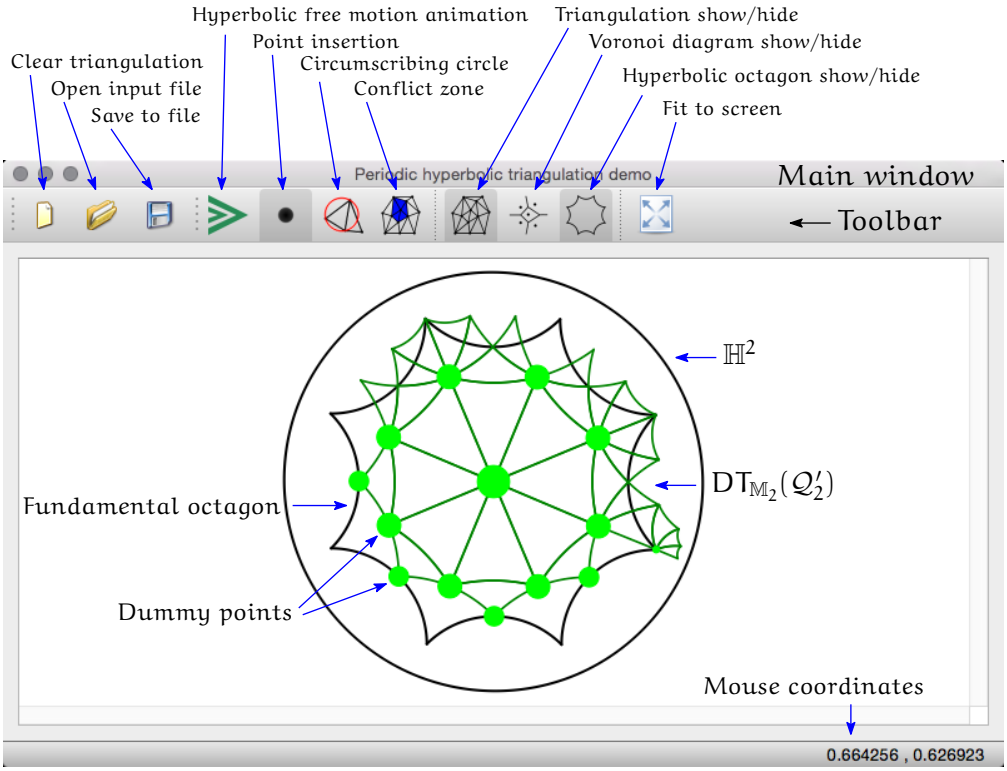

Figure IV.14: Screenshot of the Graphical User Interface (GUI) of the demo for periodic hyperbolic triangulations.

- **Hyperbolic free motion animation:** discussed in more detail later in this section;
- **Point insertion:** insert points by mouse clicks;
- **Circumscribing circle:** show the circumcircle of the canonical representative in which the mouse cursor is located (recall Figure III.13);
- **Conflict zone:** show the set of canonical faces in conflict with the mouse cursor (recall Figure III.10);
- **Triangulation show/hide:** toggle the visibility of the triangulation edges;
- **Voronoi diagram show/hide:** toggle the visibility of the dual objects of the Delaunay triangulation (see Figure IV.15);
- **Hyperbolic octagon show/hide:** toggle the visibility of the fundamental octagon;
- **Fit to screen:** recenter the view and fit the Poincaré disk.

When the display of the conflict zone and/or the circumscribing circle are active, each motion of the mouse causes the image to refresh and provides a dynamic visualization of the periodicity of the triangulation.

The demo currently does not support the removal of chosen vertices from the triangulation. The removal functionality is illustrated via the (automatic) removal of unnecessary dummy points after each insertion of a new point in the triangulation.

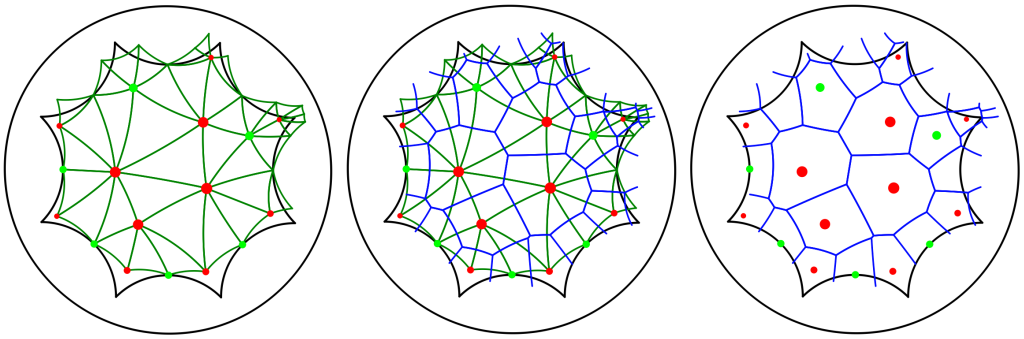

Figure IV.15: The figure shows a Delaunay triangulation with both dummy points and input points (left), the same Delaunay triangulation with the dual objects of its faces and edges (center), and just the dual objects of the Delaunay triangulation (right).

Apart from the basic actions that we described above, we also implement an idea borrowed from a paper by Balazs and Voros [BV86]. The authors work on the Bolza surface (which is not named in their work) to study chaos in hyperbolic spaces. The setup is the following. Consider a particle that executes *free motion*[18] on the Bolza surface. As the authors say, *"classical motion in this model is at the same time extremely chaotic and somehow integrable"* [BV86, pg. 112]. An interpretation of this statement is that the equations of motion of the particle on the Bolza surface give a chaotic system that can be solved numerically. Due to the lack of external forces, the trajectory of the particle is a geodesic path on the surface. Recall that geodesics on a surface are geodesics in its universal cover that are transported on the surface via its atlas. So, to follow the trajectory of the particle on the surface, one may in fact follow the geodesic trajectory of the particle in the universal cover of the surface, or the geodesic trajectory of the particle in the fundamental domain of the surface by taking into account how its sides are glued together.

Balazs and Voros define a discretization of the fundamental octagon for the Bolza surface and apply a numerical method to solve the chaotic system that describes the motion of the particle by considering periodic boundary conditions. Tracing the trajectory of the particle for a finite amount of time produces a finite segment in $\mathbb{H}^2$ that intersects finitely many copies of D_2 under the action of Γ_2. The intersections of this trajectory with each copy of D_2 can all be translated back to D_2. Balazs and Voros produce such results, and an example is shown in Figure IV.16.

In our demo application, we similarly show the trajectory of a point moving on a

[18] Free motion is the motion that a body executes in the absence of external forces, but in the presence only of an inertial force.

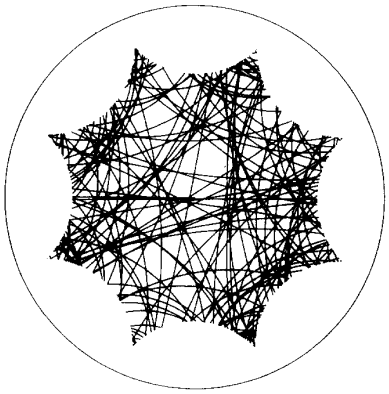

Figure IV.16: Free motion on the Bolza surface [BV86, Figure 13]. The figure shows 200 pieces of the trajectory of a moving particle superimposed on D_2.

hyperbolic segment inside the fundamental octagon. The process starts by selecting two random points in the unit disk, one as "source" p_s and the other as "target" p_t. These two points define an oriented hyperbolic line $\mathcal{L}_{[p_s,p_t]}$. We find the intersections of $\mathcal{L}_{[p_s,p_t]}$ with the octagon D_2, and choose p to be the intersection closest to the source p_s. We then start moving p along either the Euclidean segment or the Euclidean circular arc that supports the hyperbolic segment $[p_s, p_t]$, until p exits D_2. Since now p lies in a copy of D_2 that is adjacent to D_2, there exists a unique hyperbolic translation T in $\mathcal{N}_2$ that translates p back into D_2. We apply T to p, p_s, and p_t, and continue moving Tp on the segment $[Tp_s, Tp_t]$. We repeat the process indefinitely, refreshing the GUI after each step, which creates an animation until the user requests to stop the process. A sample animation captured directly from the demo can be seen at the following address:

https://tinyurl.com/bolza-free-motion

Screenshots from the process are shown in Figure IV.17.

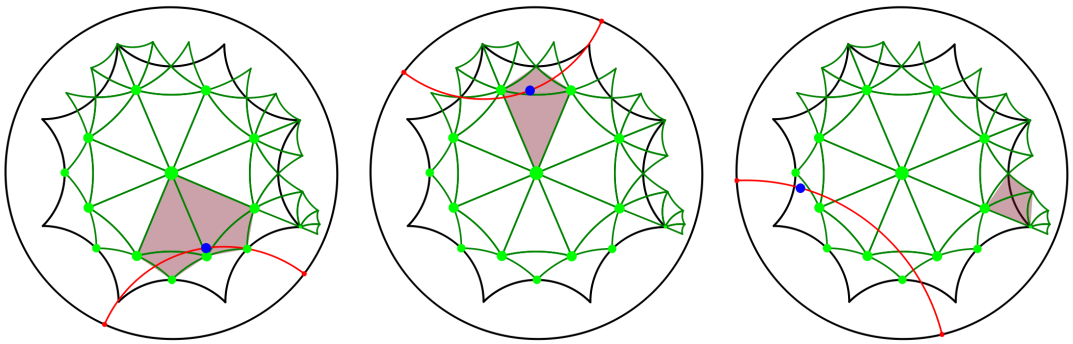

Figure IV.17: Screenshots from the demo application showing the free motion of a particle on the surface.

IV.5 Quantitative analysis of the source code

As we mentioned at the beginning of this chapter, our implementation is currently under revision for integration in CGAL [IT]. Our main contribution to the library is summarized in ten C++ header files, each one representing a separate object. We have used the tool `cloc` [Dan] to count the lines of code in these files. The output of the tool is reported in the following listing. Each row shows the filename, the number of blank lines in that file, the number of comment lines, and the number of actual code lines. The bottom row reports the sum of these quantities for all files in the list.

File	blank	comment	code
Periodic_4_hyperbolic_triangulation_2.h	237	51	770
Periodic_4_hyperbolic_Delaunay_triangulation_traits_2.h	174	48	495
Periodic_4_hyperbolic_Delaunay_triangulation_2.h	137	48	472
Hyperbolic_octagon_translation.h	56	19	164
Periodic_4_hyperbolic_triangulation_face_base_2.h	33	25	79
Periodic_4_hyperbolic_triangulation_vertex_base_2.h	22	20	66
internal/Hyperbolic_octagon_translation_word.h	92	25	385
internal/Dehn_hyperbolic_octagon_translation_word.h	60	47	213
internal/Periodic_4_hyperbolic_triangulation_dummy_14.h	50	32	189
internal/Hyperbolic_octagon_translation_matrix.h	40	19	82
SUM:	901	334	2915

We provide full documentation for our code in the form of a user manual and a reference manual to be included in the official documentation of the CGAL library online. The user manual contains a general presentation of Delaunay triangulations of the Bolza surface. It gives a brief introduction to hyperbolic geometry, explains the data structure representing a triangulation of the Bolza surface, introduces the basic objects, and gives a simple example of use of the package. The reference manual describes the concepts and models included in the package. It provides detailed descriptions of each class, its properties and methods, as well as the relationships between them. Both the user manual and the reference manual are in HTML format, produced with the help of Doxygen and CGAL scripts. In printed version, the user manual is 8 pages long, while the reference manual is 39 pages long.

The demo that we discussed in Section IV.4 is another contribution to the CGAL library. The output of the `cloc` tool for the demo is shown in the listing below. To implement the GUI for the demo, we use the `Qt` library. Without going into details, remark that the code for the free motion application is contained in the file `include/internal/hyperbolic_free_motion_animation.h` and amounts to 166 lines.

File	blank	comment	code
include/internal/Qt/TriangulationGraphicsItem.h	102	36	393
Periodic_4_hyperbolic_Delaunay_triangulation_2.ui	0	0	327
Periodic_4_hyperbolic_Delaunay_triangulation_2_demo.cpp	116	37	315

include/internal/hyperbolic_free_motion_animation.h	45	48	166
include/internal/Qt/TriangulationCircumcircle.h	28	1	107
include/internal/Qt/VoronoiGraphicsItem.h	34	22	91
include/internal/Qt/TriangulationPointInput.h	26	2	75
include/internal/Qt/HyperbolicPainterOstream.h	28	21	69
include/internal/Qt/TriangulationConflictZone.h	19	1	58
CMakeLists.txt	16	11	21
icons/about_CGAL.html	0	0	8
SUM:	414	179	1630

Finally, as all the other packages in CGAL, our code is complemented with a *testsuite*, which is a collection of programs that test various functions of the code to verify its stability and correctness.

IV.6 Experimental results

We perform tests that concern insertion times, the number of random points needed to remove all dummy points, and minimal sets of points that define a Delaunay triangulation of the Bolza surface.

Insertion time. In this experiment, we measure the time efficiency of our implementation against the time efficiency of existing CGAL triangulations. Our time efficiency test is executed on two different machines, whose technical characteristics are given in Table IV.18.

Specification	**Machine 1**	**Machine 2**
Machine type	MacBook Pro (2015)	Dell Vostro 5471 (2018)
CPU	Intel Core i5, 2.9 GHz	Intel Core i5, 1.6 GHz (up to 3.4 GHz)
RAM	16 GB, 1867 MHz	8GB, 2400 MHz
OS	MacOS X (10.10.5)	Ubuntu 18.04 (kernel 4.15.0)
Compiler	clang-700.1.81	gcc version 7.3.0

Table IV.18: Technical characteristics of the two machines used in our experiments.

We proceed in the following way. Consider a point generator that yields random points uniformly distributed in the unit disk with respect to the Euclidean metric. Using a simple accept-reject algorithm, we generate 1 million points that lie inside the half-open octagon $\mathscr{D}_2$. We insert this set of 1 million points in three triangulations:

- a periodic hyperbolic Delaunay triangulation with `CORE::Expr` as number type
- a Euclidean Delaunay triangulation with `CORE::Expr` as number type,
- and a Euclidean Delaunay triangulation with `double` as number type.

We record the insertion time for each type of triangulation. This experiment is repeated

10 times, and the insertion times are averaged over these executions. The results are reported in Table IV.19.

	Machine 1	Machine 2
Non-periodic Euclidean DT (`double`)	1 sec.	1 sec.
Non-periodic Euclidean DT (`CORE::Expr`)	24 sec.	20 sec.
Periodic hyperbolic DT (`CORE::Expr`)	55 sec.	45 sec.

Table IV.19: Runtimes for the insertion of 1 million random points in the half-open octagon $\mathscr{D}_2$.

Minimal number of random points needed to remove all dummy points. We start this experiment with a triangulation of the Bolza surface with the set of rational dummy points $\mathcal{Q}_2'$ (recall Section III.2.1). Consider a random points generator that yields random points uniformly distributed in the unit disk with respect to the Euclidean metric. We insert random points in the triangulation one by one, and after each insertion we remove the unnecessary dummy points, if there are any. As soon as the last dummy point has been removed, we stop the process and record the number of random points inserted. We repeat this experiment 1000 times, and produce the histogram shown in Figure IV.20.

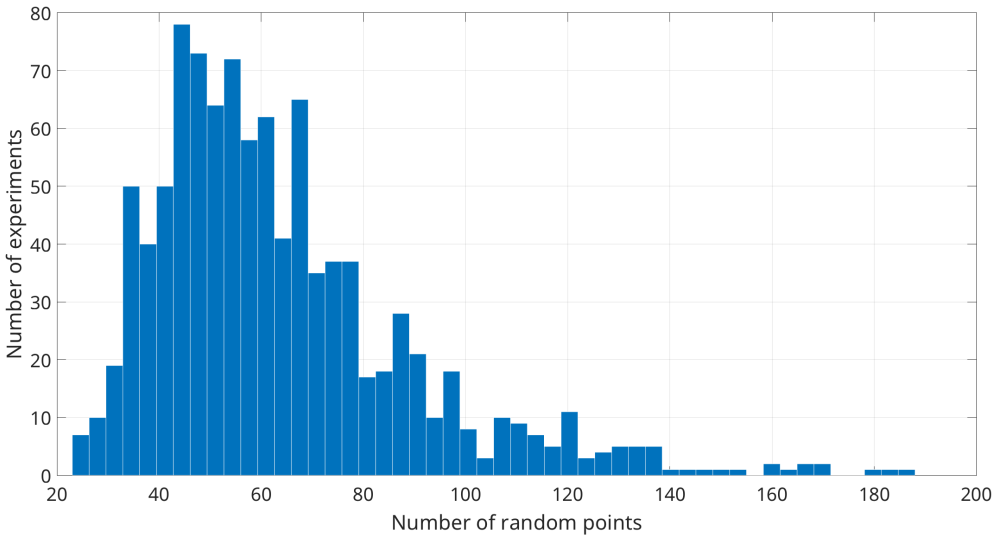

Figure IV.20: Histogram of the number of random points needed to remove all dummy points. The x-axis shows the number of random points inserted. The y-axis shows the number of experiments for which that number of random points was needed. The figure shows results for 1000 executions.

The conclusion that we draw from Figure IV.20 is that in most cases all the dummy

points can be removed from the triangulation after the insertion of 30-70 random points. We can also see that there are some cases in which over 180 random points are needed, but such cases are very rare. So a triangulation of the Bolza surface can be cleared of dummy points with a relatively low number of random points in general.

Minimal sets of points defining a Delaunay triangulation of the Bolza surface. In this experiment, we start with a triangulation of the Bolza surface with the set of dummy points $\mathcal{Q}_2'$, and we insert random points until all dummy points have been removed. We then start removing vertices from the triangulation one by one in the order in which they have been inserted until no more vertices can be removed (recall Section III.3.2). At that point, we are left with a *minimal* set of vertices that define a Delaunay triangulation of the Bolza surface, in the sense that no vertex can be removed without violating the validity condition. Each one of these sets can be used to initialize a triangulation of the Bolza surface. We execute this experiment 100 times, and produce the histogram shown in Figure IV.21.

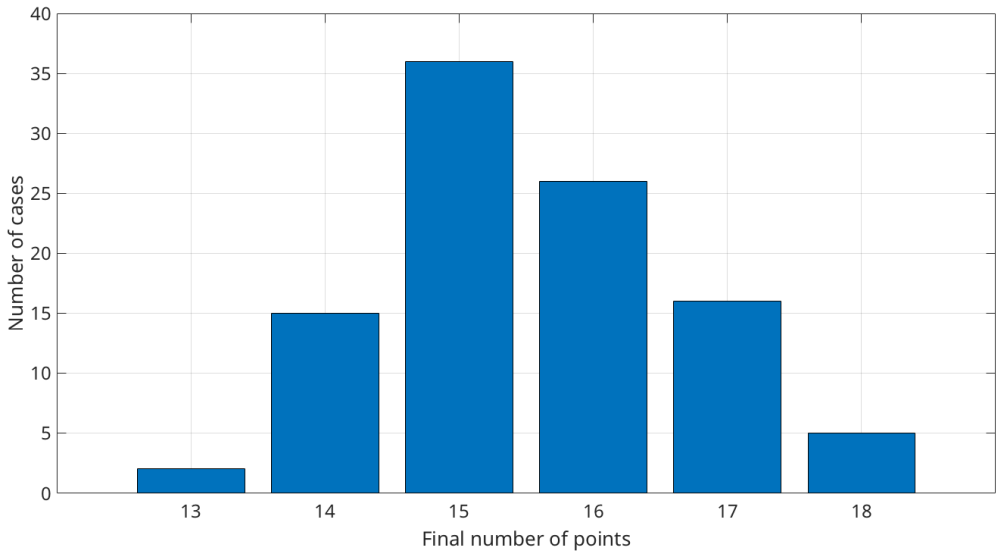

Figure IV.21: Histogram of the cardinality of minimal sets of points that define a Delaunay triangulation of the Bolza surface. The x-axis shows the number of points in the final set. The y-axis shows the number of experiments for which that final number of points was left. The figure shows results for 100 executions.

The conclusion that we draw from Figure IV.21 is that there exist minimal sets of points that define a Delaunay triangulation of the Bolza surface whose cardinality is smaller than 14, the cardinality of the set $\mathcal{Q}_2$. We find empirically sets with 13 points that define a Delaunay triangulation of the surface. See Figure IV.22 We also find cases with more points, predominantly 15 and up to 18. The advantage of using the set

$\mathcal{Q}_2$, even if its cardinality is not minimal, is that it preserves the symmetries of the fundamental octagon of $\mathbb{M}_2$.

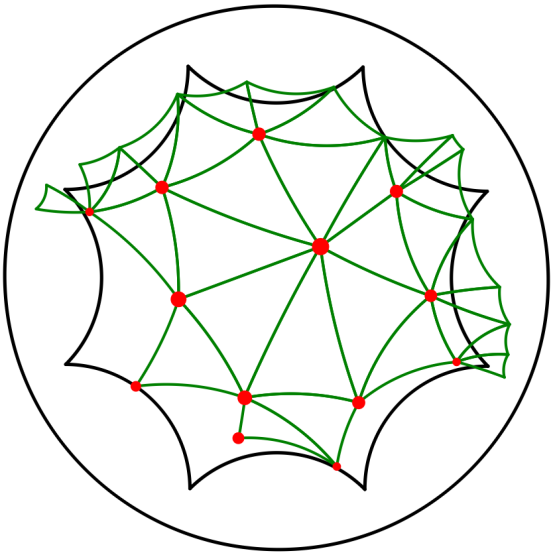

Figure IV.22: Delaunay triangulation of the Bolza surface with 13 points.

Chapter V

Delaunay triangulations of symmetric hyperbolic surfaces

We defined symmetric hyperbolic surfaces in Section II.1.6, and in Chapter III we discussed in detail the adaptation of Bowyer's incremental algorithm to the Bolza surface, the most symmetric hyperbolic surface of genus 2. In this chapter we discuss an extension of this algorithm to symmetric hyperbolic surfaces of any genus $g \geq 2$. The definitions given in this chapter are reminiscent of the definitions given in Chapter III, while the mathematical aspects presented here are fruits of our collaboration with Gert Vegter and Matthijs Ebbens [EITV, EV]. Results reported in this chapter have been presented at the 9th International Conference on Curves and Surfaces in Arcachon, France [EITV18a, EITV18b].

Throughout the whole chapter, we use the definitions given in Section II.1.6 for the symmetric hyperbolic surface $\mathbb{M}_g$ of genus $g \geq 2$, its fundamental domain D_g, and its fundamental group Γ_g. A Delaunay triangulation of $\mathbb{M}_g$ is defined as in Definition II.27 and is subject to the validity condition (II.36). In order to satisfy the validity condition for a finite set of input points $\mathcal{P}$, we follow an algorithm similar to the one for the Bolza surface:

a) Initialize a triangulation of $\mathbb{M}_g$ with a set of dummy points $\mathcal{Q}_g$ that satisfies the validity condition (II.36);
b) Insert the points of the input set $\mathcal{P}$ in the triangulation;
c) Remove any unnecessary dummy points from the triangulation.

V.1 Representation of the triangulation

In order to have a single representative of the orbit of each point under Γ_g, we introduce the *original domain* $\mathscr{D}_g \subset D_g$ for $\mathbb{M}_g$ in a similarlylar way as for the Bolza surface. To define $\mathscr{D}_g$, we first introduce an enumeration for the vertices and midpoints of the sides of D_g. Figure V.1 illustrates these elements for $g = 3$ and $g = 4$. Let V_0 be the

vertex of D_g such that the angle between the real axis and $[O, V_0]$ is $-\frac{\pi}{4g}$. The vertices V_k, $k = 1, \dots, 4g-1$ of D_g are the rotations of V_0 by $\frac{k\pi}{2g}$ around the origin. Let M_k be the hyperbolic midpoint of the side $[V_k, V_{k+1}]$ of D_g. The original domain $\mathscr{D}_g$ of $\mathbb{M}_g$ consists of the interior of D_g, the sides $[V_k, V_{k+1}]$ for $k = 2g, 2g+1, \dots, 4g-1$, and the vertex V_0. $\mathscr{D}_g$ contains exactly one representative of each point on $\mathbb{M}_g$ under the action of Γ_g, so without loss of generality we consider that all input points lie in $\mathscr{D}_g$.

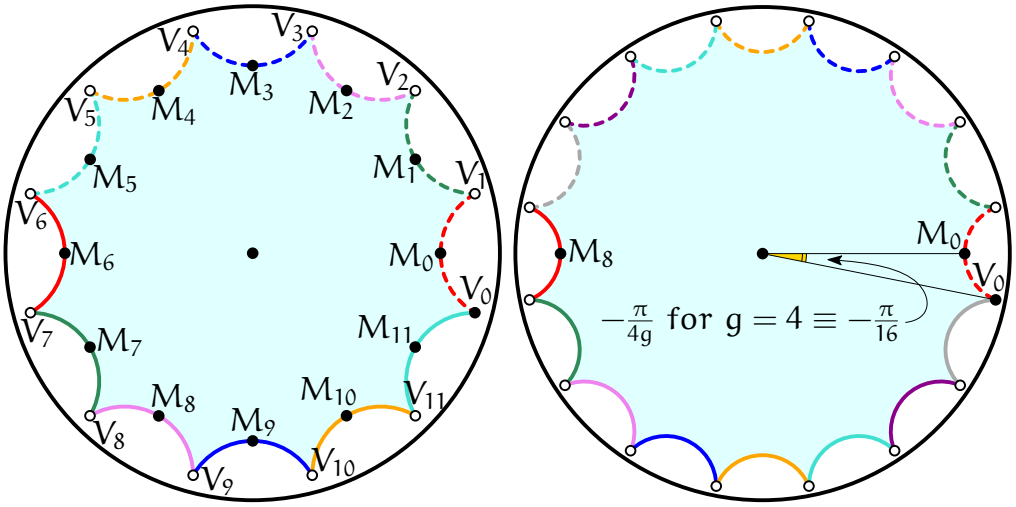

Figure V.1: Original domain $\mathscr{D}_g$ of the surface $\mathbb{M}_g$ for $g = 3$ (left) and $g = 4$ (right). $\mathscr{D}_g$ contains the interior of D_g, its sides $[V_k, V_{k+1}]$, $k = 2g, \dots, 4g-1$, and its vertex V_0.

V.1.1 Canonical representatives

We begin this section with the definition of admissible faces. Recall that $\delta(\sigma)$ denotes the circumdiameter of a face σ.

Definition V.1 (Admissible face)**.** Let $\mathcal{P} \subset \mathbb{H}^2$ a set of points that does not necessarily satisfy the validity condition (II.36). A face σ in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{P})$ is called *admissible* if $\delta(\sigma) < \frac{1}{2}\,\mathrm{sys}(\mathbb{M}_g)$.

Let $\mathcal{S}$ be a set of points $\mathcal{S} \subset \mathbb{H}^2$ that satisfies the validity condition. By Definition II.27, the projection of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{S})$ on $\mathbb{M}_g$ is a simplicial complex, and it is the Delaunay triangulation $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$ of $\mathbb{M}_g$ defined by $\mathcal{S}$. Let $\boldsymbol{\sigma}$ be a face in $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$. Each representative σ of $\boldsymbol{\sigma}$ in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{S})$ is admissible. In order to define a unique canonical representative of $\boldsymbol{\sigma}$ in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{S})$, we give two theoretical results that generalize Theorem III.2 to symmetric surfaces of genus higher than 2. First, we introduce necessary notation.

We adopt the same abuse of notation as for the Bolza surface: T denotes both a translation in Γ_g and the image of O under T. TD_g denotes the image of D_g under T.

We define the set of neighboring translations

$$\mathcal{N}_g = \left\{ T \in \Gamma_g \mid TD_g \bigcap D_g \neq \emptyset \right\}.$$

We can easily verify that $\mathcal{N}_g$ has $8g(2g-1)+1$ elements: each vertex of D_g has $4g$ polygons, and D_g has $4g$ vertices, totaling $16g^2$ polygons; D_g has been counted $4g$ times, so we need to subtract $4g-1$; each polygon sharing a side with D_g has been counted twice, therefore we need to further subtract $4g$ elements, which gives the final result. The set $\mathcal{N}_g$ is naturally ordered counter-clockwise around O, following the boundary of D_g. Recall Figure III.3 for the case of the Bolza surface. Each element T of $\mathcal{N}_g \setminus \mathrm{Id}$ is assigned an index $\mathrm{idx}_{\mathcal{N}_g}(T)$ in this sequence, while no index is defined for the identity translation. We choose $T^{\text{first}} = T_0 T_1^{-1} \dots T_{2g-2} T_{2g-1}^{-1}$ as the first element for the sequence $\mathcal{N}_g$, i.e., $\mathrm{idx}_{\mathcal{N}_g}\left(T^{\text{first}}\right) = 0$. Note that the element T^{first} is exactly the first half of the relation of Γ_g (seen as a word on the alphabet defined by the generators of Γ_g), and $T^{\text{first}} D_g$ is the reflection of D_g in the vertex V_0. We define the *Dirichlet neighborhood* $D_{\mathcal{N}_g}$ of D_g as the union

$$D_{\mathcal{N}_g} = \bigcup_{T \in \mathcal{N}_g} TD_g.$$

We present now the two results that will enable us to define the canonical representatives in $DT_{\mathbb{H}^2}(\Gamma_g \mathcal{S})$ of the faces of $DT_{\mathbb{M}_g}(\mathcal{S})$.

Lemma V.2 ([EV]). *The distance between ∂D_g and $\partial D_{\mathcal{N}_g}$ is larger than $\frac{1}{2}\mathrm{sys}(\mathbb{M}_g)$ for any $g \geq 2$, i.e.,*

$$d_{\mathbb{H}}\left(\partial D_g, \partial D_{\mathcal{N}_g}\right) > \tfrac{1}{2}\mathrm{sys}(\mathbb{M}_g), \qquad g \geq 2. \tag{V.3}$$

A full proof of Lemma V.2 is given in [EITV]. The proof is done by considering all possible cases for a distance-minimizing segment from ∂D_g to $\partial D_{\mathcal{N}_g}$. Such a segment can either intersect only one copy of D_g, or it can intersect multiple copies. It is proved that a distance-minimizing segment crossing multiple copies of D_g is always longer than a segment intersecting a single copy. Therefore, the distance between ∂D_g and $\partial D_{\mathcal{N}_g}$ equals the minimal distance between any two non-adjacent sides of D_g. This distance is realized between two sides of D_g that are adjacent to a third common side and is given as

$$d_{\mathbb{H}}\left(\partial D_g, \partial D_{\mathcal{N}_g}\right) = 2\,\mathrm{arcosh}\left(\cot\left(\tfrac{\pi}{4g}\right)\sin\left(\tfrac{\pi}{2g}\right)\right). \tag{V.4}$$

Finally, a direct comparison with $\frac{1}{2}\mathrm{sys}(\mathbb{M}_g) = \mathrm{arcosh}\left(1 + 2\cos\left(\frac{\pi}{2g}\right)\right)$ gives the result.

For the next proposition, recall from (II.29) that $\Delta(\mathcal{S})$ denotes the diameter of the largest empty disks in $\Gamma_g \mathcal{S}$.

Proposition V.5 (Inclusion property). *Let $\mathcal{S} \subset \mathscr{D}_g$ be a set of points such that $\Delta(\mathcal{S}) < \frac{1}{2}\operatorname{sys}(\mathbb{M}_g)$, and let σ be a face in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g\mathcal{S})$. If at least one of the vertices of σ lies in $\mathscr{D}_g$, then σ is contained in $\mathrm{D}_{\mathcal{N}_g}$.*

Proof. Each face σ in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g\mathcal{S})$ is circumscribed by an empty disk with diameter $\delta(\sigma) < \frac{1}{2}\operatorname{sys}(\mathbb{M}_g)$. The maximum edge length of σ is bounded by $\delta(\sigma)$, therefore it is less than $\frac{1}{2}\operatorname{sys}(\mathbb{M}_g)$. Due to Lemma V.2, we conclude that σ is contained in $\mathrm{D}_{\mathcal{N}_g}$. □

Now we can choose a unique canonical representative σ^c in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g\mathcal{S})$ for each face $\boldsymbol{\sigma}$ of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$. We follow Definition III.4 that we saw for the case of the Bolza surface.

Let $\boldsymbol{\sigma}$ be a face in $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$. By definition of $\mathscr{D}_g$, each vertex of $\boldsymbol{\sigma}$ has a unique preimage by π_g in $\mathscr{D}_g$, so, the set

$$\Sigma = \left\{ \sigma \in \pi_g^{-1}(\boldsymbol{\sigma}) \;\middle|\; \sigma \text{ has at least one vertex in } \mathscr{D}_g \right\} \tag{V.6}$$

contains at least one and at most three faces. Recall Figure III.5 for the case $g = 2$. When Σ contains only one face, then this face is completely included in $\mathscr{D}_g$, and we naturally choose it to be σ^c. Let us now assume that Σ contains two or three faces. From Proposition V.5, each face $\sigma \in \Sigma$ is contained in $\mathrm{D}_{\mathcal{N}_g}$. So, for each vertex v of σ, there is a unique translation $T(v, \sigma)$ in $\mathcal{N}_g$ such that v lies in $T(v, \sigma)\mathscr{D}_g$.

As usual, all faces in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g\mathcal{S})$ are oriented counterclockwise. For $\sigma \in \Sigma$, we denote as v_σ^{out} the first vertex of σ (in the counterclockwise order) that is not lying in $\mathscr{D}_g$. Using the indexing on $\mathcal{N}_g$ defined above, we choose σ^c as the face of Σ whose first vertex v_σ^{out} lying outside $\mathscr{D}_g$ is "closest" to the region $T^{\mathrm{first}}\mathrm{D}_g$ in the counterclockwise order around O:

Definition V.7 (Generalized canonical representative). With the notation defined above, the canonical representative of a face $\boldsymbol{\sigma}$ of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$ is the face $\sigma^c \in \Sigma$ such that

$$\operatorname{idx}_{\mathcal{N}_g}\left(T(v_{\sigma^c}^{\mathrm{out}}, \sigma^c)\right) = \min_{\sigma\in\Sigma} \operatorname{idx}_{\mathcal{N}_g}\left(T(v_\sigma^{\mathrm{out}}, \sigma)\right).$$

V.1.2 Data structure

To represent triangulations of the surface $\mathbb{M}_g$, we use a data structure similar to the data structure used for the Bolza surface, in which a triangulation is represented via its vertices and faces. Recall Figure IV.4 for an illustration. Each vertex stores an input point in $\mathscr{D}_g$ and gives access to one of its incident faces. Each face σ gives access to its incident vertices and adjacent faces, and also stores three translations $T(v_j, \sigma)$, $j = 0, 1, 2$ from the set $\mathcal{N}_g$ such that applying $T(v_j, \sigma)$ to the point stored in v_j for $j = 0, 1, 2$ produces the canonical representative σ^c of σ.

We represent translations in the following way. Consider the alphabet $\mathcal{A} = [T_0, T_1, \ldots, T_{4g-1}]$ defined by the set of generators of Γ_g. Each element of Γ_g can be

seen both as a word on the alphabet $\mathcal{A}$ and as a matrix in the form (II.14). As we have seen in Chapters III and IV, both forms are useful: seen as a word, a translation in $\mathcal{N}_g$ can be represented uniquely without expensive computations, while its matrix form is necessary to compute the images of input points. Dehn's algorithm, as we have described it for the Bolza surface in Section III.1.2, is easily generalized to the surface $\mathbb{M}_g$: the only essential change that needs to be done is to work with indices modulo $4g$ instead of modulo 8. We recall the steps of the algorithm in the sequel.

Let $\mathcal{R}_g^\infty$ be the infinite word consisting of infinitely many concatenated copies of the relation $\mathcal{R}_g = T_0 T_1^{-1} \dots T_{2g-2} T_{2g-1}^{-1} T_0^{-1} T_1 \dots T_{2g-2}^{-1} T_{2g-1}$ of Γ_g, and let $\left(\mathcal{R}_g^\infty\right)^{-1}$ be the inverse of $\mathcal{R}_g^\infty$. Let w be a non-trivial word on $\mathcal{A}$. We denote the length of w with $\ell_{\mathcal{A}}(w)$.

The first step of Dehn's algorithm is free reduction, which consists in removing all sub-words of the form TT^{-1} or $T^{-1}T$ for $T \in \mathcal{A}$. Free reduction is done by exploiting the fact that $T_j^{-1} = T_{j+2g}$.

The second step of the algorithm is the replacement of subsequences of $\mathcal{R}_g^\infty$ or $\left(\mathcal{R}_g^\infty\right)^{-1}$ in w with shorter equivalent words. This is performed by detecting a factorization of the freely-reduced word w of the form $w = w_\lambda w_\mu w_\kappa$, where $w_\mu t$ is a permutation of $\mathcal{R}_g$ or $\mathcal{R}_g^{-1}$ for some word t on $\mathcal{A}$ with $\ell_{\mathcal{A}}(t) < \ell_{\mathcal{A}}(w_\mu)$. Then $\ell_{\mathcal{A}}(w_\mu) > \ell_{\mathcal{A}}(\mathcal{R}_g)/2 = 2g$ and w_μ can be substituted in w by t^{-1}, which yields the word $w_\lambda t^{-1} w_\kappa$ with length shorter than $\ell_{\mathcal{A}}(w)$. This step exploits the fact that a sequence of letters $\left(T_{k_j}\right)_{j=0,1,\dots,n}$, $T_{k_j} \in \mathcal{A}$, is a sub-word of $\mathcal{R}_g^\infty$ of length n if, for every j from 0 to $n-1$, $k_{j+1} = (k_j + 2g + 1) \bmod 4g$. Similarly, $\left(T_{k_j}\right)$ is a sub-word of $\left(\mathcal{R}_g^\infty\right)^{-1}$ of length n if for every j from 0 to $n-1$, $k_{j+1} = (k_j - 2g - 1) \bmod 4g$.

The two steps (free reduction and relation simplification) are repeated until $w = \mathrm{Id}$ or until w cannot be further reduced. If at the end of this process the resulting word has length $\ell_{\mathcal{A}}(\mathcal{R}_g)/2 = 2g$ and is a sub-word of $\left(\mathcal{R}_g^\infty\right)^{-1}$, then w^{-1} is returned; in all other cases, we return w.

In the implementation of Dehn's algorithm that we discussed in Section IV.3.3 the generators of Γ_2 are encoded as integers. The same encoding can be used for the generators of Γ_g for $g \geq 2$, by parameterizing all operations by the genus g as described above. Since the data structure to represent words in the set $\mathcal{N}_2$ (the class `Hyperbolic_octagon_translation_word` described in Section IV.3.3) is adapted to the Bolza surface, we use a different representation for words in $\mathcal{N}_g$ for $g > 2$: we represent a word $w = \prod_{j=0}^{m} T_{k_j}$ on $\mathcal{A}$ as the ordered sequence of integers $[k_0, k_1, \dots, k_m]$. If the sequence is empty, then the translation is the identity of Γ_g.

V.2 Computation of sets of dummy points

In order to construct an initial Delaunay triangulation of $\mathbb{M}_g$, we need a set of dummy points $\mathcal{Q}_g$ that satisfies condition (II.36). For the Bolza surface, we used a set of dummy points proposed by Bogdanov *et al.*, but for the symmetric surface $\mathbb{M}_g$ with $g > 2$

we will need to *generate* suitable sets of dummy points $\mathcal{Q}_g$. The idea that we follow is borrowed from the field of mesh generation and is called *Delaunay refinement* [Rup95]. While a triangulation is the subdivision of a space defined by a fixed set of points, in mesh generation points may be added (i.e., the mesh is refined) or removed (the mesh is simplified) in order to satisfy certain criteria of the final result. Such criteria may be minimum or maximum angle of the triangles, minimum or maximum edge length, ratios of such quantities, and so on. Triangles that do not satisfy these criteria are refined by inserting their circumcenter. Similarly, edges that do not satisfy the criteria are refined by inserting their midpoints.

For the generation of dummy points we propose three methods, two of which are based on face refinement via the insertion of their circumcenters, and the third is based on edge splitting, though not by inserting the midpoints of edges. The general idea that we propose consists in starting from the set of *Weierstrass points* $\mathcal{W}_g$ for $\mathbb{M}_g$ (which we discuss in the next section). These points capture the symmetries of the surface $\mathbb{M}_g$, however the faces of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$ are non-admissible, so we refine the triangulation until all faces become admissible. We discuss in Sections V.2.3 to V.2.5 the three refinement strategies that we propose. Each strategy offers different desirable properties of the end result. At each refinement step we add finitely many points to $\mathcal{W}_g$. With each of the three strategies we obtain in finite time a set of points Q containing $\mathcal{W}_g$ for which all the faces in the triangulation $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g Q)$ are admissible. The set of points in the original domain $\mathcal{Q}_g = Q \bigcap \mathscr{D}_g$ is then a set of dummy points.

Before we discuss practical aspects of the generation of dummy points, we discuss the initial set of Weierstrass points $\mathcal{W}_g$.

V.2.1 The initial set – Weierstrass points

The set of dummy points proposed in [BTV16] captures all the symmetries of the fundamental octagon for the Bolza surface. For $\mathbb{M}_g$ with $g \geq 2$, we start the generation of dummy points with the set $\mathcal{W}_g$ of Weierstrass points, which similarly captures the symmetries of D_g.

The rigorous definition of Weierstrass points goes beyond the scope of this thesis; the interested reader can find more information on this topic, for example, in [FK92]. Without going into detail, the surfaces of the family $\mathbb{M}_g$ for $g \geq 2$ are *hyperelliptic surfaces*, and a hyperelliptic surface has exactly $2g + 2$ Weierstrass points [FK92, Section III.7]. Intuitively, these points can be understood in the following way. Consider Figure V.2, in which we have taken a double torus and "skewered" it. We rotate the double torus around this skewer by an arbitrary angle, and all points of the surface are moved, except for the "skewer" points, which are left invariant under the rotation; these are the Weierstrass points.

One property of Weierstrass points is that the minimal distance between two such points is equal to $\frac{1}{2}\,\mathrm{sys}(\mathbb{M}_g)$ [Bav92]. In the sequel we will abusively refer to the images

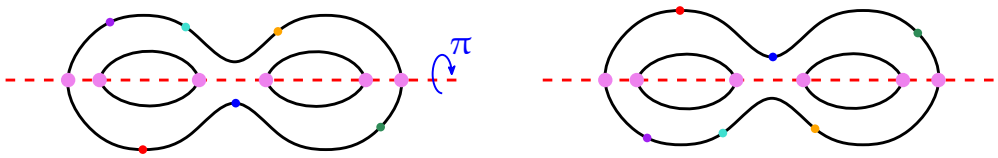

Figure V.2: Illustration of Weierstrass points on a double torus. A few points on the surface have been marked to illustrate the rotation by π around the "skewer". For g = 2, as shown on the figure, $\mathbb{M}_g$ has 6 Weierstrass points.

of the Weierstrass points of $\mathbb{M}_g$ in its original domain as the set of Weierstrass points of the surface. For the Bolza surface, the set of Weierstrass points consists of the origin O, the vertex V_0 of $\mathscr{D}_2$, and the midpoints of its closed sides $M_k, k = 4, 5, 6, 7$. For the surface $\mathbb{M}_g$, the set of Weierstrass points is similar: it consists of the origin O, the vertex V_0 of $\mathscr{D}_g$, and the midpoints M_k of its closed sides for $k = 2g, 2g+1, \ldots, 4g-1$. It should be noted that the minimal distance between two Weierstrass points of $\mathbb{M}_g$ is realized between two neighboring midpoints M_k and M_{k+1} [EITV]. In other words,

$$d_{\mathbb{H}}(M_k, M_{k+1}) = \tfrac{1}{2}\,\mathrm{sys}(\mathbb{M}_g), \qquad g \geq 2, \qquad k = 0, 1, \ldots, 4g-1. \tag{V.8}$$

Computing Weierstrass points of $\mathbb{M}_g$

To compute the coordinates of Weierstrass points, we will need to use a couple of hyperbolic trigonometric identities that we illustrate with the help of the right triangle (O, M_k, V_{k+1}) shown in Figure V.3. We name the lengths of its sides a, b, c as shown in the figure, and their opposite angles α, β, and γ, respectively. Note that apart from being right, the triangle (O, M_k, V_{k+1}) is also isosceles: the angles α and γ are both equal to $\frac{\pi}{4g}$ by construction, hence, by the Hyperbolic Sine Rule, the lengths a and c are also equal.

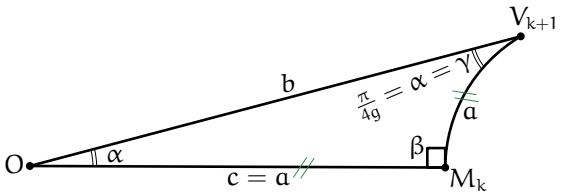

Figure V.3: We use the right triangle (O, M_k, V_{k+1}) to illustrate trigonometric rules for the computation of Weierstrass points.

Computing the coordinates of V_k. The vertices V_k all lie on a circle centered at the origin, so it suffices to compute $d_{\mathbb{E}}(O, V_0)$, which we will achieve by first computing $d_{\mathbb{H}}(O, V_0)$, and then converting it to the Euclidean metric. To compute b, we have the

identity [Bea83, Theorem 7.11.3]

$$\cosh(b) = \cot(\alpha)\cot(\gamma).$$

We substitute $\alpha = \gamma = \frac{\pi}{4g}$ and we get

$$b = d_{\mathbb{H}}(O, V_k) = \operatorname{arcosh}\left(\cot^2\left(\tfrac{\pi}{4g}\right)\right). \tag{V.9}$$

To convert $d_{\mathbb{H}}(O, V_0)$ into $d_{\mathbb{E}}(O, V_0)$, since one of the points is the origin, we can use the following formula from [Bea83, p. 131]:

$$d_{\mathbb{E}}(O, V_k) = \frac{\exp(d_{\mathbb{H}}(O, V_k)) - 1}{\exp(d_{\mathbb{H}}(O, V_k)) + 1}. \tag{V.10}$$

We substitute (V.9) into (V.10) and, after some simplifications with the help of MAPLE, we end up with

$$d_{\mathbb{E}}(O, V_k) = \sqrt{\cos\left(\tfrac{\pi}{2g}\right)}.$$

The vertices V_k can be obtained by rotating the point $\left(\sqrt{\cos\left(\frac{\pi}{2g}\right)}, 0\right)$ by $\frac{(2k-1)\pi}{4g}$ for $k = 0, 1, \ldots, 4g-1$.

Computing the coordinates of M_k. We proceed in the same way as for the points V_k, since the midpoints M_k also lie on a circle centered at the origin. First, we compute the hyperbolic length of the segment $[O, M_k]$, and then we convert it to the Euclidean metric. To compute $d_{\mathbb{H}}(O, M_k)$, we use the following formula from [Bea83, Theorem 7.11.3] (recall Figure V.3):

$$\cos(\alpha) = \cosh(a)\sin(\gamma).$$

We exploit the fact that $\alpha = \gamma = \frac{\pi}{4g}$ and we solve for a, so

$$a = d_{\mathbb{H}}(O, M_k) = \operatorname{arcosh}(\cot(\alpha)) = \operatorname{arcosh}\left(\cot\left(\tfrac{\pi}{4g}\right)\right).$$

We substitute in (V.10) in order to recover $d_{\mathbb{E}}(O, M_k)$ and we obtain (with the help of MAPLE)

$$d_{\mathbb{E}}(O, M_k) = \frac{\sqrt{\cot^2\left(\frac{\pi}{4g}\right) - 1} + \cot\left(\frac{\pi}{4g}\right) - 1}{\sqrt{\cot^2\left(\frac{\pi}{4g}\right) - 1} + \cot\left(\frac{\pi}{4g}\right) + 1}. \tag{V.11}$$

To simplify (V.11), we set temporarily $A = \cot\left(\frac{\pi}{4g}\right)$, so we have

$$\begin{aligned} d_{\mathbb{E}}(O, M_k) &= \frac{\sqrt{A^2-1}+A-1}{\sqrt{A^2-1}+A+1} \\ &= \frac{\sqrt{A-1}\sqrt{A+1}+(A-1)}{\sqrt{A-1}\sqrt{A+1}+(A+1)} \\ &= \frac{\sqrt{A-1}\left(\sqrt{A+1}+\sqrt{A-1}\right)}{\sqrt{A+1}\left(\sqrt{A-1}+\sqrt{A+1}\right)} \\ &= \sqrt{\frac{A-1}{A+1}}. \end{aligned}$$

Therefore, the Euclidean distance between the origin and any side midpoint M_k is

$$d_{\mathbb{E}}(O, M_k) = \sqrt{\frac{\cot\left(\frac{\pi}{4g}\right)-1}{\cot\left(\frac{\pi}{4g}\right)+1}}.$$

The midpoints M_k can be obtained by rotating the point $\left(\sqrt{\frac{\cot\left(\frac{\pi}{4g}\right)-1}{\cot\left(\frac{\pi}{4g}\right)+1}}, 0\right)$ by $\frac{k\pi}{2g}$ for $k = 0, 1, \ldots, 4g-1$.

Delaunay triangulation of the Weierstrass points

Since we always start the generation of dummy points with the set of Weierstrass points, we discuss here the structure of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$. See Figure V.4 - Left for an illustration for $g = 3$.

We begin with the faces incident to the origin. Consider the isosceles right triangle (O, M_k, V_{k+1}), for some $k \in \{0, 1, \ldots, 4g-1\}$. See Figure V.3. Since the segment $[O, V_{k+1}]$ is the hypotenuse of this triangle, $d_{\mathbb{H}}(O, V_{k+1}) > d_{\mathbb{H}}(O, M_k)$. No face in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$ can have $[O, V_{k+1}]$ as an edge, because such a face would not be Delaunay: its circumscribing disk would contain the midpoint M_k. To see this, note that the diameter of any circle passing through O and V_{k+1} is greater than $d_{\mathbb{H}}(O, V_{k+1})$, and

$$d_{\mathbb{H}}(O, V_k) \overset{(V.9)}{=} \operatorname{arcosh}\left(\cot^2\left(\frac{\pi}{4g}\right)\right) > \operatorname{arcosh}\left(1 + 2\cos\left(\frac{\pi}{2g}\right)\right) \underset{(V.8)}{\overset{(II.19)}{=}} d_{\mathbb{H}}(M_k, M_{k+1}). \qquad (V.12)$$

Therefore, the points O, M_k, and V_{k+1} cannot form a face of the Delaunay triangulation of the Weierstrass points because the disk through these three points contains M_{k+1}. So, all the faces of $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$ incident to the origin are in the form (O, M_k, M_{k+1}) for $k = 0, 1, \ldots, 4g-1$. To verify that

$$\operatorname{arcosh}\left(\cot^2\left(\frac{\pi}{4g}\right)\right) > \operatorname{arcosh}\left(1 + 2\cos\left(\frac{\pi}{2g}\right)\right),$$

recall that the inverse hyperbolic cosine function is strictly increasing, therefore it suffices to verify that the inequality holds for the arguments on both side. By using the identity $\cot^2\left(\frac{\pi}{4g}\right) = \frac{1+\cos\left(\frac{\pi}{2g}\right)}{1-\cos\left(\frac{\pi}{2g}\right)}$, the verification boils down to

$$\frac{1+\cos\left(\frac{\pi}{2g}\right)}{1-\cos\left(\frac{\pi}{2g}\right)} > 1 + 2\cos\left(\frac{\pi}{2g}\right),$$

which is true for any $g \geq 2$.

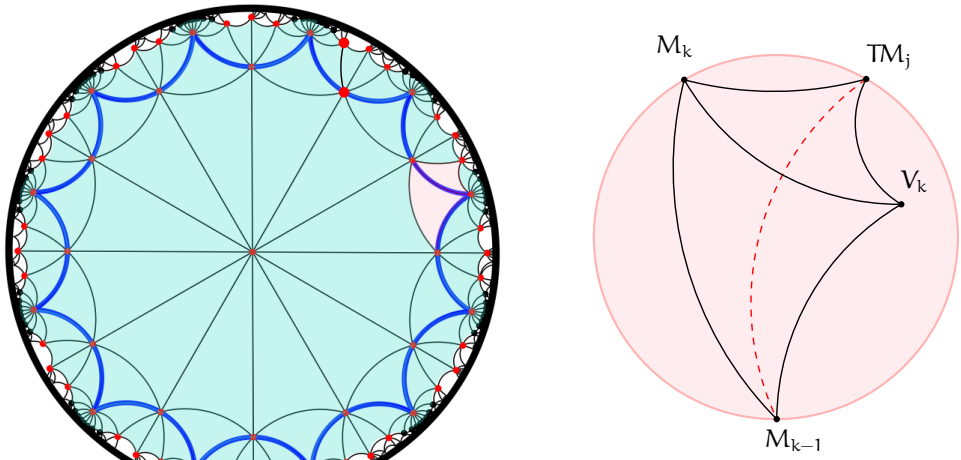

Figure V.4: ***Left:*** *The Delaunay triangulation of the set of points $\mathcal{N}_g\mathcal{W}_g$ in the hyperbolic plane for $g = 3$. The faces with at least one vertex in D_g are shaded in blue.* ***Right:*** *Zoom on the quadrilateral $(M_k, M_{k-1}, V_k, TM_j)$ with $j \in \mathbb{Z}_{4g}$ and $T \in \mathcal{N}_g$ where a flip would lead to non-Delaunay faces.*

We discuss now the faces incident to a vertex of D_g. Note that the vertices V_{k+1} of D_g are reflections of the origin in the segments $[M_k, M_{k+1}]$, since the segments $[O, M_k]$ and $[M_k, V_{k+1}]$ are equal in length. Recall that $\mathbb{H}^2$ is tiled with copies of D_g, so around a single vertex of D_g there are $4g$ copies of D_g. The reasoning we make for D_g holds in each of its copies, so all faces incident to a vertex V_{k+1} of D_g are formed by V_{k+1} and the midpoints of two adjacent sides of an image of D_g in $D_{\mathcal{N}_g}$. Note that the faces incident to the vertices of D_g are Delaunay as reflections of the faces incident to the origin: if they were not, then we could flip the edge $[M_k, M_{k+1}]$ in the quadrilateral $(O, M_k, V_{k+1}, M_{k+1})$, but as we saw before, this would lead to non-Delaunay faces. Another possibility is to flip the edge $[M_k, V_k]$ in the quadrilateral $(M_k, M_{k-1}, V_k, TM_j)$ with $j \in \mathbb{Z}_{4g}$ and $T \in \mathcal{N}_g$ such that TD_g shares the side $[V_k, V_{k+1}]$ with D_g, as shown in Figure V.4 - Right, but this would also lead to non-Delaunay faces: the hyperbolic

circle through the points M_k, M_{k-1}, and TM_j is centered at V_k, so it is not empty. Note also that all the faces in $DT_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$ with at least one vertex in D_g are contained in the Dirichlet neighborhood of D_g.

V.2.2 From Γ_g to $\mathcal{N}_g$

Up to this point, we have been talking about computing sets of dummy points by refining the triangulation $DT_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$, but in practice it is impossible to compute an infinite triangulation, even less to refine it. For a set of points Q that contains $\mathcal{W}_g$, we need to find a finite subset of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ whose projection under π_g defines a partition of the surface. We need to find a finite subset of the set of faces of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ that contains at least one image under the action of Γ_g of each piece of this partition. The identification of such a subset of faces of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ will enable us to compute and refine the Delaunay triangulation in $\mathbb{H}^2$ of finitely many images of the points Q.

Since D_g contains at least one preimage by π_g of each point on $\mathbb{M}_g$, the set of faces of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ with at least one vertex in D_g contains at least one and at most $4g$ representatives (if one of the vertices of the face is one of the vertices V_k of D_g) of the partition of $\mathbb{M}_g$ defined by the points in Q. All faces in $DT_{\mathbb{H}^2}(\Gamma_g \mathcal{W}_g)$ with at least one vertex in D_g are contained in $D_{\mathcal{N}_g}$, and we refine this triangulation until all faces become admissible; by definition, an admissible face with a vertex in D_g is also contained in $D_{\mathcal{N}_g}$. These two facts appear to suggest that, for a set of points Q that contains $\mathcal{W}_g$, the faces of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ with at least one vertex in D_g are contained in $D_{\mathcal{N}_g}$. This claim is stated in Proposition V.14, which is the main result of this section. For its proof we will need the following lemma.

Lemma V.13. *Let C_M be a Euclidean disk centered at O and passing through a point M. Let H_1 and H_2 denote the two open half-planes bounded by the Euclidean line through O and M. Let H_0 be a half-plane that contains M, bounded by another Euclidean line passing through O but not through M. Let $R_i = (H_0 \bigcap H_i) \setminus C_M, i = 1, 2$, and let $p_i \in R_i, i = 1, 2$. The disk $C(p_1, p_2)$ through O, p_1, and p_2 contains M.*

Proof. Consider the disk C_M centered at O and passing through M. Let F be the intersection of the disk with diameter $[O, M]$ with the half-plane H_1, as shown in Figure V.5 - Left. For any point $p \in H_2 \setminus C_M$, the circle through O, M, and p has a non-empty intersection with H_1, which is completely included in F.

It is easy to verify that there exist pairs of points $(p_1, p_2) \in R_1 \times R_2$ for which the point M lies inside the disk $C(p_1, p_2)$. For instance, consider a line perpendicular to the line through O and M so that M is closer to O than their intersection point, as shown in Figure V.5 - Right. If p_1 lies on this perpendicular line and p_2 is the reflection of p_1 in the line through O and M, then the disk $C(p_1, p_2)$ contains M. Since this disk varies continuously when (p_1, p_2) ranges over $R_1 \times R_2$, it is sufficient to prove that there are no pairs $(p_1^*, p_2^*) \in R_1 \times R_2$ for which M lies on the boundary of $C(p_1^*, p_2^*)$.

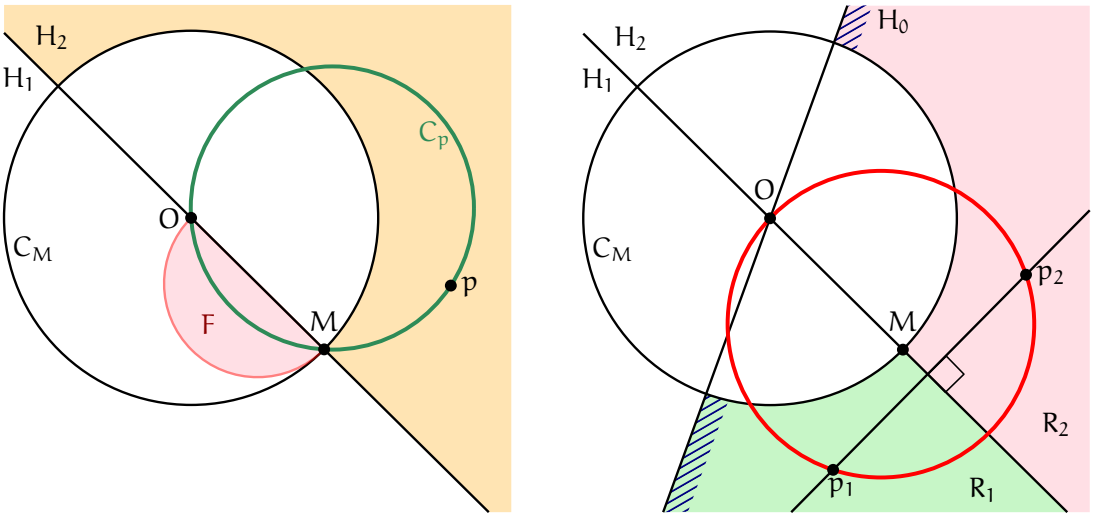

Figure V.5: Illustrations for the proof of Lemma V.13.

Assume that such points do exist, i.e., assume that there exists a pair $(p_1^*, p_2^*) \in R_1 \times R_2$ for which $C(p_1^*, p_2^*)$ is a disk with M and O on its boundary. By the observation for Figure V.5 - Left, $C(p_1^*, p_2^*)$ intersects H_1 inside the disk with diameter $[O, M]$. By a symmetric observation, $C(p_1^*, p_2^*)$ also intersects H_2 inside the same disk. Therefore, $C(p_1^*, p_2^*)$ is the disk with diameter $[O, M]$. This implies that both p_1 and p_2 lie in the disk C_M, which is a contradiction. Therefore, there exists no pair $(p_1^*, p_2^*) \in R_1 \times R_2$ for which $C(p_1^*, p_2^*)$ has M on its boundary. □

Note that Lemma V.13 is stated in Euclidean geometry, but the result can be directly used in the Poincaré disk because hyperbolic circles are represented as Euclidean circles, and hyperbolic geodesics through the origin O are supported by Euclidean lines.

Let Q be a finite set of points that contains $\mathcal{W}_g$, and let Σ_Q be the union of all faces in $DT_{\mathbb{H}^2}(\Gamma_g Q)$ with at least one vertex in $\mathscr{D}_g$:

$$\Sigma_Q = \bigcup \{ \sigma \in DT_{\mathbb{H}^2}(\Gamma_g Q) : \sigma \text{ has at least one vertex in } D_g \}.$$

Proposition V.14. *For any finite set of points Q containing $\mathcal{W}_g$ the set Σ_Q is contained in $D_{\mathcal{N}_g}$.*

Proof. We will prove that the set Σ_Q is included in $D_{\mathcal{N}_g}$ by showing that all edges in $DT_{\mathbb{H}^2}(\Gamma_g Q)$ with one endpoint in D_g have their other endpoint inside $D_{\mathcal{N}_g}$.

Let e be a segment with an endpoint in D_g and an endpoint outside $D_{\mathcal{N}_g}$. We will prove that no disk passing through the endpoints of e is empty. There are two cases to consider: e either crosses only one image of D_g under Γ_g in $D_{\mathcal{N}_g} \setminus D_g$, or it crosses several of its images. We examine each case separately.

Case A: The edge e only crosses one image of D_g before leaving $D_{\mathcal{N}_g}$.

This case is illustrated in Figure V.6 - Left. Let $TD_g, T \in \mathcal{N}_g$ be the Dirichlet region that e crosses. The image $e' = T^{-1}e$ of e then crosses D_g, intersecting two of its non-adjacent sides S_i and S_j in the points p_i and p_j, respectively. Let M_i be the midpoint of S_i, and M_j the midpoint of S_j. The sides S_i and S_j can be either two opposite sides of D_g or not.

Case 1: S_i *and* S_j *are opposite sides of* D_g. We distinguish two cases.

> ***Case i:*** *the hyperbolic segment through* $[p_i, p_j]$ *passes through the origin.* Trivially, any disk through p_i, p_j contains the origin.
>
> ***Case ii:*** *the hyperbolic segment through* $[p_i, p_j]$ *does not pass through the origin.* There exists a line l_O through O that has both p_i and p_j on the same side. Let M_ℓ me the midpoint of some side of D_g on the same side of l_O as the points p_i and p_j. Consider the circle centered at O and passing through M_ℓ, the line l_O, and the line through O and M_ℓ. By Lemma V.13, the disk $C(p_i, p_j)$ passing through O, p_i, and p_j contains M_ℓ. Since O and M_ℓ are on both sides of the segment $[p_i, p_j]$, any disk through p_i and p_j contains either M_ℓ or O, therefore there is no empty disk that passes through p_i and p_j.

Case 2: S_i *and* S_j *are not opposite sides of* D_g. In this case, there exists a line l_O through O for which S_i and S_j are on the same side. Let M_ℓ be the midpoint of a side between S_i and S_j on the same side of l_O. Consider the disk centered at O that passes through M_ℓ (and, of course, through all the other midpoints M_k as well), and consider also the line through O and M_ℓ. By Lemma V.13, the disk $C(p_i, p_j)$ passing through O, p_i, and p_j contains M_ℓ. Since O and M_ℓ are on both sides of the segment $[p_i, p_j]$, any disk through p_i and p_j contains either M_ℓ or O, therefore there is no empty disk that passes through p_i and p_j.

So, in both cases 1 and 2, there is no empty disk that passes through p_i and p_j. Assume now that there is an empty disk that passes through the endpoints of e'. This empty disk can then be shrunk continuously so that it passes through p_i and p_j. The shrunk version of the disk must be also empty, which is a contradiction. Therefore, there is no empty disk passing through the endpoints of e', which implies that e' (and, by consequence, e) cannot be an edge in $DT_{\mathbb{H}^2}(\Gamma_g Q)$.

Case B: The edge e crosses several images of D_g before leaving $D_{\mathcal{N}_g}$.

This case is illustrated in Figure V.6 - Right. There exist multiple images of e in $DT_{\mathbb{H}^2}(\Gamma_g Q)$ that intersect D_g, in fact as many as the number of Dirichlet regions it intersects. Each one of these images intersects two adjacent sides of D_g. Let e' be an image of e that intersects two adjacent sides S_i and S_ℓ of D_g so that the hyperbolic line supporting e' separates O and the midpoint M_ℓ. Note that such an image of e exists always: e either separates O and the midpoint M_ℓ, or it separates an image of O under some translation T of Γ_g and M_ℓ; in the second case, $T^{-1}e$ separates O and the midpoint $T^{-1}M_\ell$. The edge e' intersects also the side S_j (adjacent to S_ℓ) of the Dirichlet region

that shares the side S_ℓ with D_g. Let p_i and p_j be the intersection points of e' with S_i and S_j, respectively. Consider the circle centered at the origin that passes through M_ℓ. Consider also the line through O and M_ℓ and the line through O perpendicular to it. By Lemma V.13, the disk $C(p_i, p_j)$ passing through O, p_i, and p_j contains M_ℓ. Since O and M_ℓ are on both sides of the segment $[p_i, p_j]$, any disk through p_i and p_j contains either M_ℓ or O, therefore there is no empty disk that passes through p_i and p_j. Now, assume that there is an empty disk that passes through the endpoints of e'. This empty disk can then be shrunk continuously so that it passes through p_i and p_j, and it will still be empty, which is a contradiction. Therefore, there is no empty disk passing through the endpoints of e', which implies that e' (and, by consequence, e) cannot be an edge in $DT_{\mathbb{H}^2}(\Gamma_g Q)$.

In conclusion, no edge of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ can have an endpoint in D_g and an endpoint outside $D_{\mathcal{N}_g}$, therefore all faces with at least one vertex in D_g are included in $D_{\mathcal{N}_g}$. □

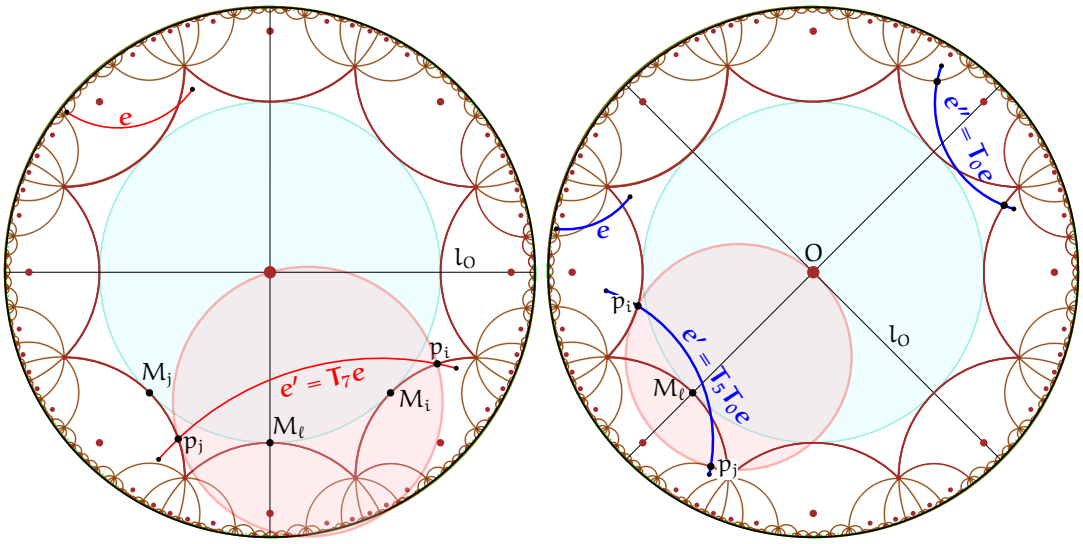

Figure V.6: Illustration for the proof of Proposition V.14 for $g = 2$.

For any set of points Q that contains the Weierstrass points $\mathcal{W}_g$, the projection of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ by π_g defines a partition of $\mathbb{M}_g$. Due to Proposition V.14, the Delaunay triangulation of the set $\mathcal{N}_g Q$ is built from a finite subset of the set of faces of $DT_{\mathbb{H}^2}(\Gamma_g Q)$ and contains at least one image of each piece of this partition of $\mathbb{M}_g$. Based on this result, two of the strategies that we propose consist in computing $DT_{\mathbb{H}^2}(\mathcal{N}_g \mathcal{W}_g)$ and then refining the triangulation until the validity condition is satisfied. If Q is the final set of points that contains $\mathcal{W}_g$, then $\mathcal{N}_g Q \bigcap \mathscr{D}_g$ is a suitable set of dummy points.

V.2.3 Sequential strategy

The sequential strategy is the simplest of the three refinement strategies. Its correctness is based upon the following result.

Proposition V.15. *For a given set of points* Q *that contains the set of Weierstrass points* $\mathcal{W}_g$*, each face* σ' *in* $DT_{\mathbb{H}^2}(\Gamma_g Q)$ *has at least one image* σ *under some translation* T *in* Γ_g *such that the circumcenter of* σ *is in* D_g*. Moreover,* σ *is a face of* $DT_{\mathbb{H}^2}(\mathcal{N}_g Q)$*.*

Proof. Assume that there exists no image of σ' in $DT_{\mathbb{H}^2}(\Gamma_g Q)$ whose circumcenter is in D_g. This implies that there exists a point in $\mathbb{H}^2$ (the circumcenter of σ') that has no image in D_g under the action of Γ_g, which is a contradiction.

Let σ be an image of σ' in $DT_{\mathbb{H}^2}(\Gamma_g Q)$ whose circumcenter c_σ is in D_g. Assume that σ lies outside $D_{\mathcal{N}_g}$. For this to happen, $\delta(\sigma)$ must satisfy $\delta(\sigma) \geq d_{\mathbb{H}}(\partial D_g, \partial D_{\mathcal{N}_g})$. Since $Q \supseteq \mathcal{W}_g$, we have that $\delta(\sigma) \leq \Delta(\mathcal{W}_g)$. We know that

$$\Delta(\mathcal{W}_g) < 2d_{\mathbb{H}}(O, M_k) = 2\,\mathrm{arcosh}\left(\cot\left(\tfrac{\pi}{4g}\right)\right) < 2\,\mathrm{arcosh}\left(\cot\left(\tfrac{\pi}{4g}\right)\sin\left(\tfrac{\pi}{2g}\right)\right) \overset{(V.4)}{=} d_{\mathbb{H}}(\partial D_g, \partial D_{\mathcal{N}_g}). \tag{V.16}$$

From (V.16), $\delta(\sigma) < d_{\mathbb{H}}(\partial D_g, \partial D_{\mathcal{N}_g})$, therefore the circumcenter of σ cannot lie in D_g if σ lies outside $D_{\mathcal{N}_g}$. By consequence, the image σ in $DT_{\mathbb{H}^2}(\Gamma_g Q)$ of σ' whose circumcenter is in D_g is in fact a face of $DT_{\mathbb{H}^2}(\mathcal{N}_g Q)$. □

Proposition V.15 enables us to formulate the sequential strategy because it implies that, for a set of points Q containing $\mathcal{W}_g$, we can always find a representative of any face of $DT_{\mathbb{M}_g}(\Gamma_g Q)$ in $DT_{\mathbb{H}^2}(\mathcal{N}_g Q)$ whose circumcenter is in D_g. The sequential strategy is summarized in the following algorithm.

```
Algorithm: SequentialRefinement
```

```
Input:  Set of Weierstrass points W_g.
Output: Set of dummy points Q_g.
```

```
Q_g ← W_g ;
Do
  T ← DT_{H^2}(N_g Q_g) ;
  Δ ← max{ δ(σ) | σ is a face of T,  σ has at least one vertex in D_g } ;
  If ( Δ ≥ (1/2) sys(M_g) ) Then
    σ ← face of T with δ(σ) = Δ and circumcenter in D_g ;
    c_σ ← circumcenter of σ ;
    Q_g ← Q_g ⋃ {c_σ} ;
  End_if
While ( Δ ≥ (1/2) sys(M_g) ) ;

Return Q_g ;
```

Snippet V.7: Sequential refinement strategy for the generation of dummy points

One of the downsides of the sequential strategy is that the resulting point sets are not symmetric. An example for genus 3 is given in Figure V.8. Note that these figures have been obtained from a preliminary implementation of the sequential refinement strategy. We discuss our preliminary implementation in detail in Section V.5. Let us mention here that our implementation computes successfully dummy points with the sequential strategy for genus 2 and 3: for higher genus, our code crashes due to insufficient accuracy (with up to $2048 \times g$ precision bits).

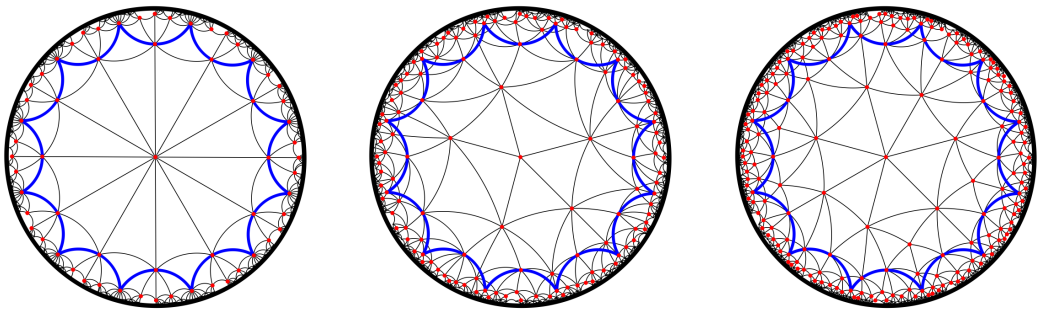

Figure V.8: Construction of dummy points with the sequential strategy for $g = 3$. ***Left:*** *Initial triangulation $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{W}_g)$.* ***Center:*** *Intermediate refinement step.* ***Right:*** *Final triangulation $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$.*

Proposition V.17. *The sequential algorithm terminates. The resulting dummy point set $\mathcal{Q}_g$ satisfies the validity condition (II.36) and has cardinality of order $\Theta(g)$.*

A proof of Proposition V.17 is given in [EITV]. To prove that the algorithm terminates, a circle packing argument is used: the circumdiameter of any face in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{W}_g)$ is larger than $\frac{1}{2}\operatorname{sys}(\mathbb{M}_g)$, and at each step we add to $\mathcal{W}_g$ the circumcenter of a face of maximal circumdiameter. For any two distinct inserted circumcenters, the two disks centered at them with diameters $\frac{1}{2}\operatorname{sys}(\mathbb{M}_g)$ are found to be disjoint. The area of such disks is fixed for a fixed g, as is the area of $\mathbb{M}_g$, so we can add only finitely many circumcenters until we cover the whole surface. By consequence, the algorithm terminates.

The termination of the algorithm is subject to the `Do-While` loop in Snippet V.7, which ends only when all faces with at least one vertex in D_g have circumdiameters smaller than $\frac{1}{2}\operatorname{sys}(\mathbb{M}_g)$. Therefore, the algorithm outputs a set of points that satisfies (II.36).

Finally, the cardinality of $\mathcal{Q}_g$ is proved to be of order $\Theta(g)$ by using a circle packing argument on disks of diameter $\frac{1}{4}\operatorname{sys}(\mathbb{M}_g)$ that cover $\mathbb{M}_g$. Note that, with the sequential refinement strategy, the size of the resulting sets of dummy points is asymptotically minimal [JR80].

V.2.4 Sequential strategy with symmetries

This strategy follows the same logic as the sequential and aims to maintain the rotational symmetry of D_g. The idea is that instead of inserting one circumcenter at a time, we also insert all its rotations by $\frac{k\pi}{2g}$, $k = 1, \dots, 4g-1$ around the origin. Note that this strategy also uses Proposition V.15. It is summarized in the following steps:

```
Algorithm: SequentialRefinementWithSymmetries
```

```
Input:  Set of Weierstrass points W_g.
Output: Set of dummy points Q_g.
```

$\mathcal{Q}_g \leftarrow \mathcal{W}_g$;
Do
  $\mathcal{T} \leftarrow \mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g \mathcal{Q}_g)$;
  $\Delta \leftarrow \max\{\, \delta(\sigma) \mid \sigma$ is a face of $\mathcal{T}$, σ has at least one vertex in $D_g \,\}$;
  If ($\Delta \geq \frac{1}{2}\,\mathrm{sys}(\mathbb{M}_g)$) Then
    $\sigma \leftarrow$ face of $\mathcal{T}$ with $\delta(\sigma) = \Delta$ and circumcenter in D_g;
    $C_0 \leftarrow$ circumcenter of σ;
    For k from 1 to $4g-1$ Do
      $C_k \leftarrow \mathtt{Rotate}(C_{k-1}, \frac{\pi}{2g})$;
    End_for
    $\mathcal{Q}_g \leftarrow \mathcal{Q}_g \bigcup \{\, C_k,\ k = 0, \dots, 4g-1 \,\}$;
  End_if
While ($\Delta \geq \frac{1}{2}\,\mathrm{sys}(\mathbb{M}_g)$);

Return $\mathcal{Q}_g$;

Snippet V.9: Sequential refinement strategy with symmetries for the generation of dummy points

See Figure V.10 for an illustration of the set of dummy points that it yields for $g = 3$. These figures are also obtained with our preliminary implementation. The preliminary code successfully computes sets of dummy points for genus 2 and 3 with the sequential strategy with rotations, while for higher genus it crashes due to insufficient accuracy (with up to $2048 \times g$ precision bits).

Proposition V.18. *The symmetric algorithm terminates. The resulting dummy point set satisfies the validity condition* (II.36) *and has cardinality of order* $\Theta(g \log g)$.

The proof for Proposition V.18 is also given in [EITV]. The first two claims follow directly from the proof of Proposition V.17. The result that the cardinality of the resulting set of dummy points is of order $\Theta(g \log g)$ is obtained from the facts that the algorithm terminates in $\Theta(\log(g))$ steps (based on a circle packing argument similar to the one used in the proof of Proposition V.17), and that at each step $4g$ points are added to the point set.

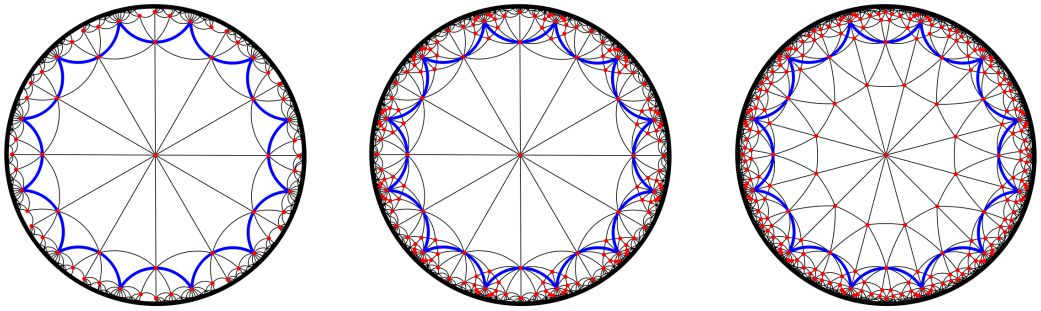

*Figure V.10: Construction of dummy points with the sequential strategy with symmetries for $g = 3$. **Left:** Initial triangulation $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{W}_g)$. **Center:** First refinement step. **Right:** Second refinement step. This is the final triangulation $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$.*

V.2.5 Deterministic strategy

The deterministic strategy is a bit more complicated than the previous two. We propose this strategy because it preserves the symmetry of D_g and the combinatorial structure of $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ is known. The idea behind this strategy is to refine the triangles (O, M_k, M_{k+1}) by inserting points on their sides $[O, M_k]$ and $[O, M_{k+1}]$. The algorithm is the following.

As a first step, the midpoints of the segments $[M_k, M_{k+1}], k = 0, 1, \ldots, 4g-1$ are added to $\mathcal{W}_g$:

$$U = \{\, U_k = \mathrm{Midpoint}(M_k, M_{k+1}), \quad k = 0, 1, \ldots, 4g-1 \,\},$$
$$\mathcal{W}_g = \mathcal{W}_g \cup U$$

Then, create a set of points P lying on the segments $[O, M_k]$ such that:

$$P = \left\{ P_k^j, \quad k = 0, 1, \ldots, 4g-1, \quad j = 0, 1, \ldots, m \right\}$$
$$P_k^j \in [O, M_k], \quad d_{\mathbb{H}}\left(P_k^j, P_k^{j+1}\right) = \tfrac{1}{4}\,\mathrm{sys}(\mathbb{M}_g).$$

The index m is such that the circumdiameter of $\left(O, P_k^m, P_{k+1}^m\right)$ is smaller than $\frac{1}{2}\,\mathrm{sys}(\mathbb{M}_g)$ for $k = 0, 1, \ldots, 4g-1$. We then reflect the points P in the segments $[M_k, M_{k+1}]$ to obtain the set of reflected points R:

$$R = \left\{ R_k^j, \quad k = 1, \ldots, 4g-1, \quad j = 0, 1, \ldots, m \right\}$$
$$R_k^j \text{ is the reflection of } P_k^j \text{ in } [M_k, M_{k+1}] \text{ for } k = 0, 1, \ldots, 2g-1,$$
$$R_k^j \text{ is the reflection of } P_k^j \text{ in } [M_{k-1}, M_k] \text{ for } k = 2g, 2g+1, \ldots, 4g-1.$$

The distinction for the segments in which we reflect the points P_k^j is done to avoid duplicate points. By construction, all the faces in the Delaunay triangulation of these points

are admissible [EITV]. The idea for the proof of this claim is given in Section V.3.1, where we will also discuss the "manual" construction of the Delaunay triangulation of $\mathbb{M}_g$ defined by these dummy points. The strategy is summarized in the following pseudo-code listing.

```
Algorithm: DeterministicRefinement
```

```
Input:  Set of Weierstrass points W_g.
Output: Set of dummy points Q_g; Number of subdivisions m.
```

$P \leftarrow \text{EmptySet}()$;
For k from 0 to 4g-1 Do
  $P_k^0 \leftarrow M_k$;
  $P \leftarrow P \bigcup \{ P_k^0 \}$;
End_for
$P_{4g}^0 \leftarrow M_0$;
$U \leftarrow \text{EmptySet}()$;
For k from 0 to 4g-1 Do
  $U_k \leftarrow \text{Midpoint}(P_k^0, P_{k+1}^0)$;
  $U \leftarrow U \bigcup \{U_k\}$;
End_for
$Q_g \leftarrow W_g \bigcup \{U_k,\ k = 0, \ldots, 4g-1\}$;
$R \leftarrow \text{EmptySet}()$;
$m \leftarrow 0$;
Do
  $m \leftarrow m+1$;
  // PointOnSegment(A, B, d): returns a point
  // on the segment [A, B] at distance d from A
  $P_0^m \leftarrow \text{PointOnSegment}(P_0^{m-1}, O, \frac{1}{4}\,\text{sys}(\mathbb{M}_g))$;
  $R_0^m \leftarrow \text{Reflect}(P_0^m, [M_0, M_1])$;
  For k from 1 to 4g-1 Do
    $P_k^m \leftarrow \text{Rotate}(P_{k-1}^m, \frac{\pi}{2g})$;
    // Take cases for reflection to avoid duplicate points
    If (k < 2g) Then
      $R_k^m \leftarrow \text{Reflect}(P_k^m, [M_k, M_{k+1}])$;
    Else
      $R_k^m \leftarrow \text{Reflect}(P_k^m, [M_{k-1}, M_k])$;
    End_if
  End_for
  $P \leftarrow P \bigcup \{P_k^m,\ k = 0, \ldots, 4g-1\}$;
  $R \leftarrow R \bigcup \{R_k^m,\ k = 0, \ldots, 4g-1\}$;
While ($\text{Circumdiameter}(O, P_0^m, P_1^m) \geq \frac{1}{2}\,\text{sys}(\mathbb{M}_g)$);

$Q_g \leftarrow Q_g \bigcup P \bigcup R$;

```
Return Q_g, m;
```

Snippet V.11: Deterministic refinement strategy for the generation of dummy points

Note that the algorithm returns not only the set of dummy points, but also the number of subdivisions m, which is necessary for the initialization process that we discuss in Section V.3. The deterministic subdivision strategy is illustrated in Figure V.12. We zoom in on one of the triangles (O, M_k, M_{k+1}) to show how points are added successively on its sides.

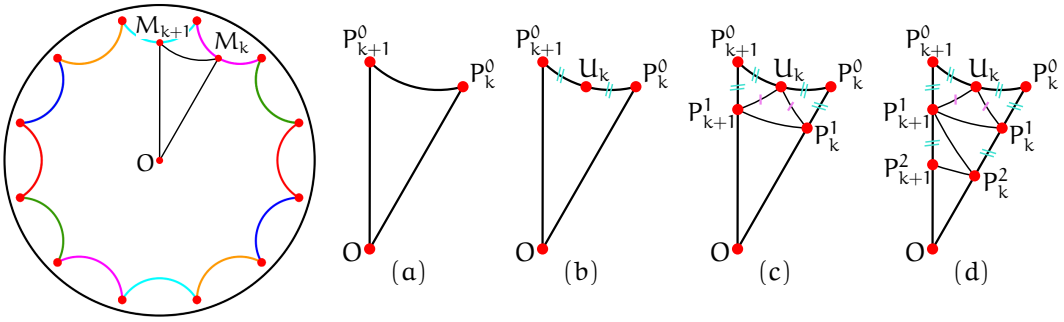

*Figure V.12: Construction of dummy points with the deterministic strategy for $g = 3$. **Left:** Fundamental polygon for $g = 3$ with one of the triangles (O, M_k, M_{k+1}). **(a):** Zoom on the triangle (O, M_k, M_{k+1}); note that M_k and M_{k+1} have been renamed to P^0_k and P^0_{k+1}, respectively. **(b):** The midpoint U_k of P^0_k and P^0_{k+1} is added. Note that $d_{\mathbb{H}}(P^0_k, U_k) = \frac{1}{4}\operatorname{sys}(\mathbb{M}_g)$. **(c):** The points $P^1_j, j = k, k+1$ are added at distance $\frac{1}{4}\operatorname{sys}(\mathbb{M}_g)$ from P^0_j. **(d):** The points $P^2_j, j = k, k+1$ are added at distance $\frac{1}{4}\operatorname{sys}(\mathbb{M}_g)$ from P^1_j. The refinement continues until the circumdiameter of the triangle (O, P^m_k, P^m_{k+1}) becomes smaller than $\frac{1}{2}\operatorname{sys}(\mathbb{M}_g)$ for some m.*

Proposition V.19 ([EV]). *The cardinality of the set of dummy points obtained with the deterministic strategy is of order $\Theta(g \log(g))$.*

A proof of the proposition is given in [EITV]. The proof is based on the fact that the algorithm terminates in $\Theta(\log(g))$ steps, and at each step $8g$ points are added to the point set.

V.3 Construction of the triangulation

Let $\mathcal{P}$ be a set of input points in $\mathscr{D}_g$. In this section we discuss the steps to construct the Delaunay triangulation of $\mathbb{M}_g$ defined by the set of points $\mathcal{S} = \mathcal{P} \bigcup \mathcal{Q}_g$, where $\mathcal{Q}_g$ is a set of dummy points generated with any one of the strategies presented in the previous section. We begin by discussing the initialization of the triangulation of $\mathbb{M}_g$ with a set of dummy points $\mathcal{Q}_g$ generated with one of the strategies discussed in Sections V.2.3 to V.2.5, after which we discuss other operations on the triangulation.

V.3.1 Initialization

The construction of the triangulation $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ depends on the strategy that has been used to generate the set of dummy points $\mathbb{M}_g$. If $\mathcal{Q}_g$ has been generated with one of the sequential strategies, then we construct $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ by using the combinatorial structure of $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$. For sets of dummy points generated with the deterministic strategy, the combinatorial structure of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ can be described explicitly, so we do not actually compute $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$. We discuss both cases in the sequel.

Dummy points generated with the sequential strategies

The initialization of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ with dummy points generated with either one of the sequential strategies has been implemented in our preliminary code.

With both sequential strategies (recall Sections V.2.3 and V.2.4), we have no *a priori* knowledge of the combinatorial structure of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$. We therefore use the combinatorial information provided by $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ to build $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$, however we need to represent $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ with a data structure that enables us to keep track of translated images of vertices and faces. We begin by describing such an adapted data structure.

We assign a unique integer identifier to each point in $\mathcal{Q}_g$. We represent $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ via its faces and vertices. Each vertex gives access to one of its incident faces and stores the image $Tp, T \in \mathcal{N}_g$ of a point $p \in \mathcal{Q}_g$, the translation T, and the integer identifier of p. Each face of the triangulation stores the usual adjacency and incidence relations. The indices and translations stored in the vertices enable us to implicitly keep track of vertices and faces in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$: to recognize that two faces are images under the action of Γ_g, it is sufficient to verify that their vertices store the same set of integer identifiers. The set of vertices in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ that store identity translations are the *original vertices* of the triangulation that lie in the half-open original polygon $\mathscr{D}_g$.

We describe now the construction of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ from $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$.

The process starts with the creation of an empty triangulation $\mathcal{T}$ that will become $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$. For each original vertex w of $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ with identifier $k, k = 0, 1, \ldots, |\mathcal{Q}_g| - 1$, we create a new vertex v_k in $\mathcal{T}$ (stored in an ordered list at index k) that stores the same point as w. We keep the correspondence between w and v_k in a bidirectional map that will be used in the last step of the algorithm.

Next, we create the faces of $\mathcal{T}$. For each *canonical* face σ in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ (according to Definition V.7), we recover the identifiers id_j, $j = 0, 1, 2$ and the translations tr_j, $j = 0, 1, 2$ stored in its three vertices. We then create a new face τ in $\mathcal{T}$ with vertices v_{id_j}, $j = 0, 1, 2$, and store in it the three translations tr_j, $j = 0, 1, 2$. We keep the correspondence between σ and τ in a second bidirectional map.

The next step is to set face adjacency relations by using the face correspondence established in the previous step. For each face τ in $\mathcal{T}$, we recover its corresponding (original) face σ in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$. We find the canonical representative nbr_j, $j = 0, 1, 2$

in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ for each neighbor of σ, and set the neighbors of τ to be the faces in $\mathcal{T}$ corresponding to each nbr_j.

The last step is to set the face pointers for each vertex of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$. For each vertex v of $\mathcal{T}$, we recover the corresponding vertex w in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$. We find the canonical representative σ in $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$ of the face to which w points. Finally, we set v to point to the face τ in $\mathcal{T}$ that corresponds to σ.

We give pseudo-code for this process in the following Snippet V.13.

```
Algorithm: InitializePeriodicTriangulation
```

```
Input:  Delaunay triangulation DT_{H^2}(N_g Q_g).
Output: Delaunay triangulation DT_{M_g}(Q_g).
```

```
// Create an empty periodic hyperbolic triangulation
T ← CreateEmptyTriangulation();

// Create vertices, set points, keep vertex correspondence map
Vmap ← CreateEmptyBidirectionalMap();
For each Original_vertex w_k in Q_g Do
  j ← w_k.id();
  v_j ← T.CreateNewVertex();
  v_j.set_point(w_k.get_point());
  Vmap.set_pair(v_j, w_k);
End_for

Σ^c_{Q_g} ← FindAllCanonicalRepresentatives(DT_{H^2}(N_g Q_g));

// Create faces, set translations, keep face correspondence map
Fmap ← CreateEmptyBidirectionalMap();
For each Face σ_k in Σ^c_{Q_g} Do
  For j from 0 to 2 Do
    id_j ← σ_k.vertex(j).id();
    tr_j ← σ_k.vertex(j).translation();
  End_for
  τ_k ← T.CreateNewFace(v_{id_0}, v_{id_1}, v_{id_2});
  τ_k.set_translations(tr_0, tr_1, tr_2);
  Fmap.set_pair(τ_k, σ_k);
End_for

// Set adjacency relations by using face correspondence map
For each Face τ_k in DT_{M_g}(Q_g) Do
  σ_k ← Fmap.get_pair_of(τ_k);
  For j from 0 to 2 Do
    nbr_j ← σ_k.neighbor(j);
```

```
    If nbr_j is not in Σ^c_{Q_g} Then
      nbr_j ← FindCanonicalRepresentative(nbr_j, DT_{H²}(N_g Q_g));
    End_if
    τ_k.set_neighbor(Fmap.get_pair_of(nbr_j));
  End_for
End_for

// Set face pointers; use vertex and face correspondence map
For each Vertex v_k in DT_{M_g}(Q_g) Do
  w_k ← Vmap.get_pair_of(v_k);
  σ_k ← FindCanonicalRepresentative(w_k.get_face(), DT_{H²}(N_g Q_g));
  v_k.set_face(Fmap.get_pair_of(σ_k));
End_for

Return T;
```

Snippet V.13: Initialization of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ *from* $\mathrm{DT}_{\mathbb{H}^2}(\mathcal{N}_g\mathcal{Q}_g)$.

Dummy points generated with the deterministic strategy

The initialization of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ with dummy points generated with the deterministic strategy has not been implemented in our preliminary code.

A description of the algorithm is given in Snippet V.16. Note that the input to the algorithm consists of the set of dummy points as well as the number of subdivisions m during their generation. Refer to Figure V.15 for an illustration for $g = 3$ and $m = 2$. Figure V.14 shows a general indexing scheme for the faces of the triangulation.

With the deterministic strategy, we obtain a set of dummy points $\mathcal{Q}_g$ that subdivide each triangle (O, M_k, M_{k+1}) into $2(m+1)$ triangles that belong to one of six "classes" defined as follows:

a) triangles incident to the origin, in the form (O, P_k^m, P_{k+1}^m);
b) subdivision triangles in the form $(P_k^j, P_{k+1}^{j-1}, P_{k+1}^j)$, $j = 2, \ldots, m$;
c) subdivision triangles in the form $(P_k^j, P_k^{j-1}, P_{k+1}^{j-1})$, $j = 2, \ldots, m$;
d) triangles in the form (P_k^1, U_k, P_{k+1}^1);
e) triangles in the form (P_k^1, P_k^0, U_k);
f) triangles in the form $(P_{k+1}^1, U_k, P_{k+1}^0)$.

The reflection of a triangle that belongs in the class a (respectively, b, c, d, e, f) in the segment $[P_k^0, P_{k+1}^0]$ belongs in a class denoted with a' (respectively, b', c', d', e', f'). Each quadrilateral $(O, M_k, V_{k+1}, M_{k+1})$ is subdivided into $4(m+1)$ triangles that belong to a total of 12 classes. The whole $4g$-gon is subdivided into $16g(m+1)$ triangles, and each triangle is indexed by an integer in $[0, 16g(m+1))$. The indices of the triangles are assigned so that the triangles in each quadrilateral $(O, M_k, V_{k+1}, M_{k+1})$

have indices in $4k(m+1) + \mathbb{Z}_{4(m+1)}$ for $k = 0, 1, \ldots, 4g-1$. Moreover, all the triangles contained in (O, P^0_k, P^0_{k+1}) have even indices, while all the triangles contained in $(P^0_k, V_{k+1}, P^0_{k+1})$ have odd indices. The indices are assigned progressively to faces in the classes $a, a', b, b', c, c', d, d', e, e', f$, and f'.

Before we describe the algorithm to construct $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$, we give the following result.

Proposition V.20 ([EV]). *The triangles of each class* $a, b, c, d, e,$ *and* f *and their reflections* $a', b', c', d', e',$ *and* f' *are Delaunay.*

A proof of Proposition V.20 is given in [EITV]. The proof is done with an extensive case analysis.

The initialization process starts with the creation of an empty triangulation $\mathcal{T}$ that will become $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$. Recall from Section V.2.5 that the set $\mathcal{Q}_g$ is the union of the sets $P = \left\{ P^j_k,\ k = 0, \ldots, 4g-1,\ j = 0, \ldots, m \right\}$, $U = \left\{ U^j_k,\ k = 0, \ldots, 4g-1,\ j = 0, \ldots, m \right\}$ and $R = \{ R_k,\ k = 0, \ldots, 4g-1 \}$, as well as the origin O and the vertex V_0 of $\mathscr{D}_g$. For each point in P and R, we create a new vertex in $\mathcal{T}$ with the same name as the point that it stores. For the vertices R^j_k in the sides of $\mathscr{D}_g$, we consider *pointers* to vertices R^j_{k+2g} that simply refer to R^j_k, $k = 0, 1, \ldots, 2g-1$; in other words, the vertex R^j_{k+2g} and the vertex R^j_k refer to the same object that stores one original point in $\mathscr{D}_g$.

We continue with the creation of the faces of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ iteratively. We work in the quadrilateral $(O, M_k, V_{k+1}, M_{k+1})$ for $k = 0, \ldots, 4g-1$. New faces are created and stored in an ordered list at specific indices. The index at which a face is placed is considered to be its identifier. Faces are placed at the following positions:

- the face in the class a to the position $4k(m+1)$;
- the face a' to the position $4k(m+1)+1$;
- the faces in the classes b and c positions $4k(m+1)+m-j+2$ and $4k(m+1)+m-j+4$, respectively, for $j = 2, \ldots, m$;
- the faces in the classes b' and c' to positions $4k(m+1)+m-j+3$ and $4k(m+1)+m-j+5$, respectively, for $j = 2, \ldots, m$;
- the faces in the classes d, e, and f to positions $4k(m+1)+4(m-1)+2, 4k(m+1)+4(m-1)+4$, and $4k(m+1)+4(m-1)+6$, respectively;
- the faces in the classes d', e', and f' to positions $4k(m+1)+4(m-1)+3, 4k(m+1)+4(m-1)+5$, and $4k(m+1)+4(m-1)+7$, respectively.

See Figure V.14 for an illustration of the classes, and Figure V.15 for an example of the indices assigned to faces for $g = 3$ and $m = 2$.

The next step is to set adjacency relations between the faces. With the indexing given in the previous step, the process is straightforward. The indices of the neighbors of each face are reported in Snippet V.16, so we do not repeat them here.

We continue with the assignment of translations to the faces of the triangulation. The only faces that have non-identity translations are the ones incident to some of the vertices R_k^j, P_k^0, or $V_k, k \neq 0$. Vertices R_k^j on the sides $[V_k, V_{k+1}], k = 0, \ldots, 2g-1$ are translates of original vertices under the generator T_k of Γ_g. For the images of V_0, the situation is a bit more complicated: from the vertex V_j we obtain the vertex V_{j+2g+1} by applying the generator T_{j+2g+1}.[19]

The final step is to set the face pointers of the vertices of $\mathcal{T}$. For each vertex, we select one of its incident faces and set it iteratively as described at the end of Snippet V.16.

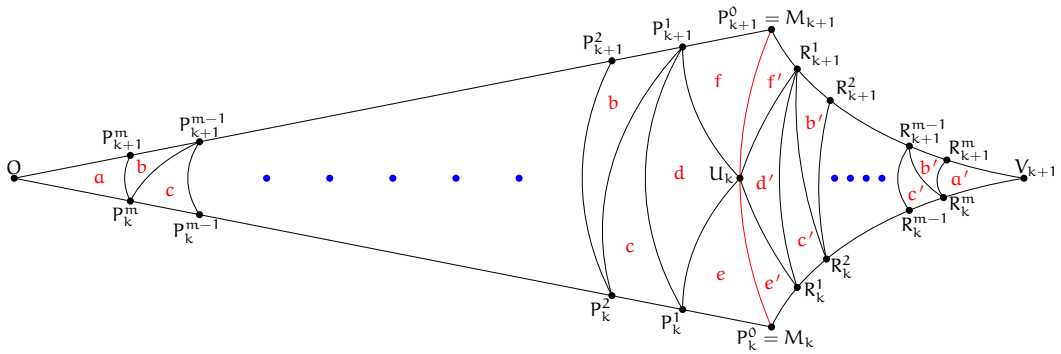

Figure V.14: Zoom on the faces of the triangulation of $\mathbb{M}_g$ defined by the deterministic dummy points.

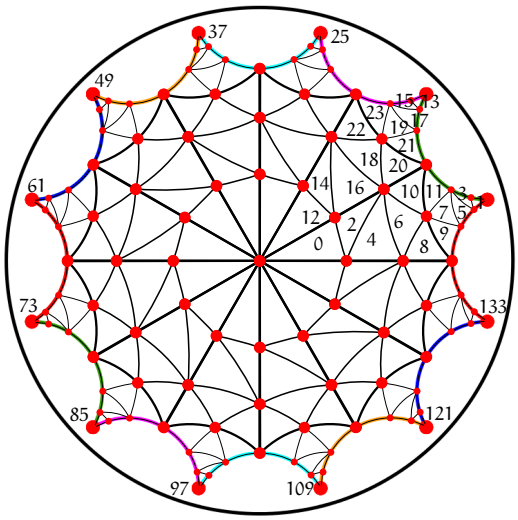

Figure V.15: Initialization of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ with deterministic set of dummy points. The number next to each vertex is the identifier of the face incident to that vertex.

[19] This series of translations gives the relation of Γ_g.

Algorithm: `InitializeDeterministicPeriodicTriangulation`

Input: Set of dummy points $\mathcal{Q}_g$, Number of subdivisions m.
Output: Delaunay triangulation $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$.

```
// Create an empty periodic hyperbolic triangulation
T ← CreateEmptyTriangulation();

// Create vertices, set points
For each Point p_k^j in P ⊂ Q_g Do
  P_k^j ← T.CreateNewVertex();
  P_k^j.set_point(p_k^j);
End_for

For each Point r_k^j in R ⊂ Q_g with k ∈ [2g, 4g − 1] Do
  R_k^j ← T.CreateNewVertex();
  R_k^j.set_point(r_k^j);
  // Create a reference to the newly created vertex
  // (to avoid complicated index expressions)
  R_{k+2g}^j ← T.ReferenceVertex(R_k^j);
End_for

// Create faces
L ← CreateEmptyList();
For k from 0 to 4g − 1 Do
  L.PutAtPosition(4k(m + 1), T.CreateFace(O, P_k^m, P_{k+1}^m));
  L.PutAtPosition(4k(m + 1) + 1, T.CreateFace(V, R_{k+1}^m, R_k^m));
  For j from 2 to m Do
    L.PutAtPosition(4k(m + 1) + m − j + 2, T.CreateFace(P_k^j, P_{k+1}^{j−1}, P_{k+1}^j));
    L.PutAtPosition(4k(m + 1) + m − j + 3, T.CreateFace(R_k^j, R_{k+1}^j, R_{k+1}^{j−1}));
    L.PutAtPosition(4k(m + 1) + m − j + 4, T.CreateFace(P_k^j, P_k^{j−1}, P_{k+1}^{j−1}));
    L.PutAtPosition(4k(m + 1) + m − j + 5, T.CreateFace(R_k^j, R_{k+1}^{j−1}, R_k^{j−1}));
  End_for
  L.PutAtPosition(4k(m + 1) + 4(m − 1) + 2, T.CreateFace(P_k^1, U_k, P_{k+1}^1));
  L.PutAtPosition(4k(m + 1) + 4(m − 1) + 3, T.CreateFace(R_k^1, R_{k+1}^1, U_k));
  L.PutAtPosition(4k(m + 1) + 4(m − 1) + 4, T.CreateFace(P_k^1, P_k^0, U_k));
  L.PutAtPosition(4k(m + 1) + 4(m − 1) + 5, T.CreateFace(R_k^1, U_k, P_k^0));
  L.PutAtPosition(4k(m + 1) + 4(m − 1) + 6, T.CreateFace(P_{k+1}^1, U_k, P_{k+1}^0));
  L.PutAtPosition(4k(m + 1) + 4(m − 1) + 7, T.CreateFace(R_{k+1}^1, P_{k+1}^0, U_k));
End_for

// Set adjacency relations
For k from 0 to 4g − 1 Do
```

```
L[4k(m+1)].set_neighbors(      L[4k(m+1)+2],
                               L[mod(4(k+1)(m+1),16g(m+1))],
                               L[mod(4(k-1)(m+1),16g(m+1))]);
L[4k(m+1)+1].set_neighbors(    L[4k(m+1)+3],
                               L[mod(4(k+2g-1)(m+1)+1,16g(m+1))],
                               L[mod(4(k+2g+1)(m+1)+1,16g(m+1))]);

For j from 2 to m Do
  L[4k(m+1)+m-j+2].set_neighbors(L[mod(4(k+1)(m+1)+m-j+4,16g(m+1))],
                                 L[4k(m+1)+m-j],
                                 L[4k(m+1)+m-j+4]);
  L[4k(m+1)+m-j+3].set_neighbors(L[mod(4(k+2g+1)(m+1)+m-j+5,16g(m+1))],
                                 L[4k(m+1)+m-j+5],
                                 L[4k(m+1)+1]);
  L[4k(m+1)+m-j+4].set_neighbors(L[4k(m+1)+m-j+6],
                                 L[4k(m+1)+m-j+2],
                                 L[mod(4(k-1)(m+1)+m-j+2,16g(m+1))]);
  L[4k(m+1)+m-j+5].set_neighbors(L[4k(m+1)+m-j+7],
                                 L[mod(4(k+2g-1)(m+1)+m-j+3,16g(m+1))],
                                 L[4k(m+1)+m-j+3]);
End_for
```

```
  L[4k(m+1)+4(m-1)+2].set_neighbors(L[4k(m+1)+4(m-1)+6],
                                    L[4k(m+1)+m-j+4],
                                    L[4k(m+1)+4(m-1)+4]);
  L[4k(m+1)+4(m-1)+3].set_neighbors(L[4k(m+1)+4(m-1)+7],
                                    L[4k(m+1)+4(m-1)+5],
                                    L[4k(m+1)+4(m-1)+1]);
  L[4k(m+1)+4(m-1)+4].set_neighbors(L[4k(m+1)+4(m-1)+5],
                                    L[4k(m+1)+4(m-1)+2],
                                    L[mod(4(k-1)(m+1)+4(m-1)+6,16g(m+1))]);
  L[4k(m+1)+4(m-1)+5].set_neighbors(L[4k(m+1)+4(m-1)+4],
                                    L[mod(4(k+2g-1)(m+1)+4(m-1)+7,16g(m+1))],
                                    L[4k(m+1)+4(m-1)+3]);
  L[4k(m+1)+4(m-1)+6].set_neighbors(L[4k(m+1)+4(m-1)+7],
                                    L[mod(4(k+1)(m+1)+4(m-1)+4,16g(m+1))],
                                    L[4k(m+1)+4(m-1)+2]);
  L[4k(m+1)+4(m-1)+7].set_neighbors(L[4k(m+1)+4(m-1)+6],
                                    L[4k(m+1)+4(m-1)+3],
                                    L[mod(4(k+2g+1)(m+1)+4(m-1)+5,16g(m+1))]);
End_for

// Set translations
For k from 0 to 2g-2 Do
  // Translations for the vertices are set in a separate loop
  L[4k(m+1)+1].set_translations(Id,T_k,T_{k+1});
  For j from 2 to m Do
    L[4k(m+1)+m-j+3].set_translations(T_k,T_{k+1},T_{k+1});
    L[4k(m+1)+m-j+5].set_translations(T_k,T_{k+1},T_k);
  End_for
  L[4k(m+1)+4(m-1)+3].set_translations(T_k,T_{k+1},Id);
  L[4k(m+1)+4(m-1)+4].set_translations(Id,Id,T_{k+1});
  L[4k(m+1)+4(m-1)+5].set_translations(T_k,Id,T_k);
  L[4k(m+1)+4(m-1)+6].set_translations(Id,T_k,Id);
  L[4k(m+1)+4(m-1)+7].set_translations(T_{k+1},T_{k+1},Id);
End_for

j ← 0;
T ← T_0;
For k from 1 to 4g-1 Do
  L[4k(m+1)+1].set_translation(0,T);
  j ← mod(j+2g+1,4g);
  T ← T·T_j;
```

```
End_for

// Set face pointer for each vertex
For k from 0 to 4g − 1 Do
  P_k^0.set_face(L[4k(m + 1) + 4(m − 1) + 4]);
  P_k^1.set_face(L[4k(m + 1) + 4(m − 1) + 4]);
  For j from 2 to m Do
    P_k^j.set_face(L[4k(m + 1) + m − j + 4]);
  End_for

  U_k.set_face(L[4k(m + 1) + 4(m − 1) + 2]);

  R_k^1.set_face(L[4k(m + 1) + 4(m − 1) + 5]);
  For j from 2 to m Do
    R_k^j.set_face(L[4k(m + 1) + m − j + 5]);
  End_for
End_for

Return T;
```

Snippet V.16: Initialization of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{Q}_g)$ *from the set of dummy points* $\mathcal{Q}_g$ *generated with the deterministic strategy.*

V.3.2 Insertion and other operations

Let $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$ be the Delaunay triangulation of $\mathbb{M}_g$ defined by a set of points $\mathcal{S}$ containing $\mathcal{Q}_g$. To insert a new point p in $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$, first the conflict zone of p must be identified. Both the conflict zone identification and the insertion processes follow the algorithms described for the Bolza surface in Sections III.2.2 and III.2.3, respectively. All the notions and definitions in these sections generalize directly to $\mathbb{M}_g$.

Moreover, the operations described in Section III.3, namely the location of points and the removal of vertices (which enables also the removal of dummy vertices), are also directly applicable to $\mathbb{M}_g$.

V.4 Predicates

We have already discussed the importance of predicates for the computation of Delaunay triangulations. In this section, we go over the predicates necessary for the computation of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$ for a set $\mathcal{S}$ that satisfies the validity condition, and then we discuss practical issues with the exact evaluation of these predicates.

The algorithm that we presented in this chapter requires the Orientation and InCircle predicates, which we discussed in Section III.4.1. The Orientation predicate is used during the Euclidean visibility walk, in order to identify a face in conflict with

an input point p. The InCircle predicate is used to test if faces are in conflict with input points.

In order to provide point location facilities, the SideOfOrientedHyperbolicSegment predicate is needed, refining the result of the Euclidean visibility walk and returning the hyperbolic triangle that contains a query point.

Lastly, for the Bolza surface we used the predicate SideOfOriginalOctagon to check whether an input point lies inside the original octagon. We introduce now a new predicate named SideOfOriginalPolygon that returns the relative position of a point with respect to the original $4g$-gon $\mathscr{D}_g$ for the surface $\mathbb{M}_g$. The predicate SideOfOriginalPolygon is evaluated as follows.

Consider p to be a point in $\mathbb{H}^2$; we want to check if p lies in $\mathscr{D}_g$ or not. First, we check if p lies in D_g, and we refine our test accordingly in special cases.

For each $k = 0, 1, \ldots, 4g-1$ we consider the position of p with respect to the segments $[O, V_k]$ and $[O, V_{k+1}]$, as shown in Figure V.17. If for some j the Orientation of p with respect to $[O, V_j]$ is non-negative, and also its orientation with respect to $[O, V_{j+1}]$ is negative, then we have identified a half-open "slice" of $\mathbb{H}^2$ that contains p. Now we consider the following cases:

- If p is inside the disk through M_j, V_j, and V_{j+1}, then p lies outside D_g, so it also lies outside $\mathscr{D}_g$;
- If p is outside the disk through M_j, V_j, and V_{j+1}, then it is inside D_g and also inside $\mathscr{D}_g$;
- Otherwise, p is on the boundary of D_g, and we need to consider the following cases:
  - ▷ If the orientation of p with respect to $[O, V_j]$ is null, then p lies both on the segment $[O, V_j]$ and on the side $[V_j, V_{j+1}]$, so $p = V_j$;
    - ∗ If $j = 0$, then p is inside $\mathscr{D}_g$;
    - ∗ Otherwise, p is outside $\mathscr{D}_g$;
  - ▷ Otherwise, if the orientation of p with respect to $[O, V_j]$ is positive:
    - ∗ If $0 \leq j \leq 2g - 1$, then p is outside $\mathscr{D}_g$;
    - ∗ Otherwise, p is inside $\mathscr{D}_g$.

So, the SideOfOriginalPolygon predicate is evaluated with at most $4g$ calls to the Orientation predicate, and one call to the InCircle predicate.

In Section III.4.2 we discussed the algebraic degree of predicates for the case of the Bolza surface. In order to obtain the reported results, we performed an extensive case analysis with the help of Maple. Unfortunately, we have not been able to perform the same analysis for symmetric hyperbolic surfaces of genus higher than 2 with the use of symbolic computations. In the following section we present the algebraic expressions

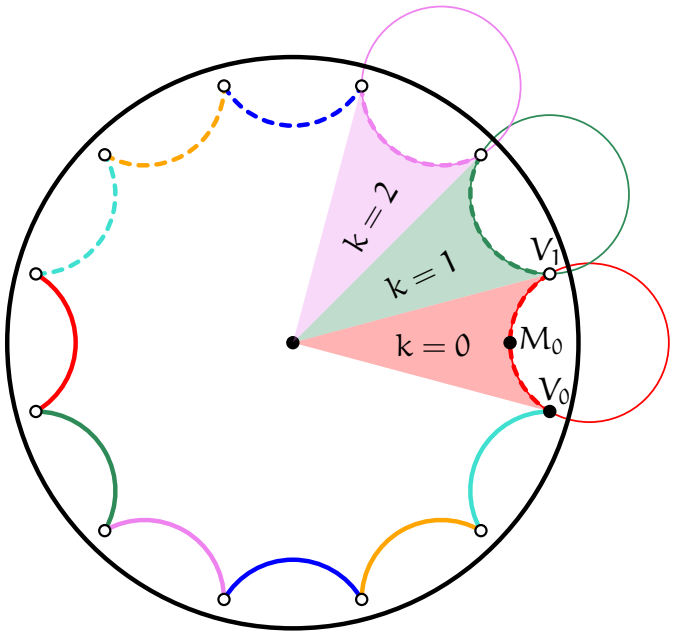

Figure V.17: Illustration of how the SideOfOriginalOctagon *predicate is evaluated.*

that appear in predicates and discuss practical issues related to their exact evaluation.

Exact representation of translation matrices

Recall from (II.17) that the matrices of the generators of Γ_g are

$$T_k = \begin{bmatrix} \cot\left(\frac{\pi}{4g}\right) & \exp\left(\frac{ik\pi}{2g}\right)\sqrt{\cot^2\left(\frac{\pi}{4g}\right) - 1} \\ \exp\left(-\frac{ik\pi}{2g}\right)\sqrt{\cot^2\left(\frac{\pi}{4g}\right) - 1} & \cot\left(\frac{\pi}{4g}\right) \end{bmatrix}, \quad k = 0, 1, \ldots, 4g-1.$$

One of the properties of Möbius transformations is that multiplying the matrix of the transformation with a number has no effect on the action of the transformation. We multiply T_k with $\sin\left(\frac{\pi}{4g}\right)$, and the matrices of the generators of Γ_g take the form

$$T_k = \begin{bmatrix} \cos\left(\frac{\pi}{4g}\right) & \exp\left(\frac{ik\pi}{2g}\right)\sqrt{\cos\left(\frac{\pi}{2g}\right)} \\ \exp\left(-\frac{ik\pi}{2g}\right)\sqrt{\cos\left(\frac{\pi}{2g}\right)} & \cos\left(\frac{\pi}{4g}\right) \end{bmatrix}, \quad k = 0, 1, \ldots, 4g-1.$$

So, the generators of Γ_g can be expressed in an exact manner if we are able to express exactly the trigonometric functions $\sin(\cdot)$ and $\cos(\cdot)$ evaluated at angles of the form $\frac{\pi}{2g}$, $g \geq 2$. We represent such numbers as roots of *Chebyshev polynomials*, which are defined via the recurrence relation

$$\begin{aligned} P_0(x) &= 1, \\ P_1(x) &= x, \\ P_n(x) &= 2xP_{n-1}(x) - P_{n-2}(x). \end{aligned}$$

Chebyshev polynomials are the unique polynomials with the property that $P_n(\cos(\theta)) = \cos(n\theta)$. By taking $\theta = \frac{\pi}{n}$, we have

$$P_n\left(\cos\left(\tfrac{\pi}{n}\right)\right) = \cos(\pi) = -1.$$

So, $\cos\left(\frac{\pi}{n}\right)$ is a root of the polynomial $P_n(x)+1$. In fact, $\cos\left(\frac{\pi}{n}\right)$ is the largest positive root of this polynomial for the following reason. Another property of Chebyshev polynomials is that $-1 \leq P_n(x) \leq 1, n \geq 0, x \in [-1,1]$, and the extrema -1 and 1 of $P_n(x)$ are attained at the points $\cos\left(\frac{k\pi}{n}\right)$, $k = 0,\ldots,n$; for $k = 1$, we obtain $P_n(x) = -1$, which gives the largest positive root of $P_n(x) + 1$. This gives us an exact representation of $\cos\left(\frac{\pi}{2g}\right)$. We represent $\sin\left(\frac{\pi}{2g}\right)$ as $\sqrt{1 - \cos^2\left(\frac{\pi}{2g}\right)}$.

Advantages of using Chebyshev polynomials are that they are very easy to construct and their roots are real, so we can try to use CORE to work with such expressions. Therefore we represent the matrices of the generators of Γ_g by using the largest positive roots of polynomials $P_{2g}(x) + 1$. The matrices of the other translations in $\mathcal{N}_g$ are computed by multiplying the matrices of the generators of Γ_g. If we assume that we are given input points with rational coordinates, then the input to the predicates are the images of such points under hyperbolic translations with algebraic coefficients.

CORE guarantees that predicates will be evaluated exactly. In practice, however, the evaluation of predicates that involve expressions as the ones discussed in the previous paragraph has a high computational cost that renders the implementation non-efficient. More precisely, the problem is manifested when two equal numbers are compared. In such cases, computations are required with increasingly higher precision until the *root separation bound*[20] is attained, so equality testing for complicated expressions is very expensive.

To avoid this phenomenon, we impose a finite precision ε to CORE. The effect is that whenever two numbers x and y are compared and $|x - y| < \varepsilon$, then the numbers are considered equal. Defining a finite precision that is sufficient to guarantee exactness without reaching the root separation bound would be a way to avoid unnecessary computations. However the definition of such precision requires an accuracy analysis depending on the genus, which we did not perform. In the next section we describe a preliminary code that we implement.

[20] The root separation bound of a polynomial is the lower bound on the distance between any two distinct roots of the polynomial [BFM+01, DSY07].

V.5 Preliminary experimental results

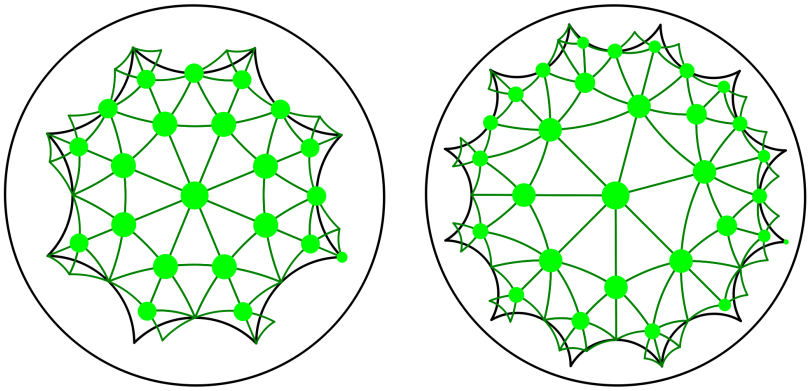

Figure V.18: Triangulations of $\mathbb{M}_g$ for $g = 2$ (left) and $g = 3$ (right) with dummy points generated with the sequential strategy.

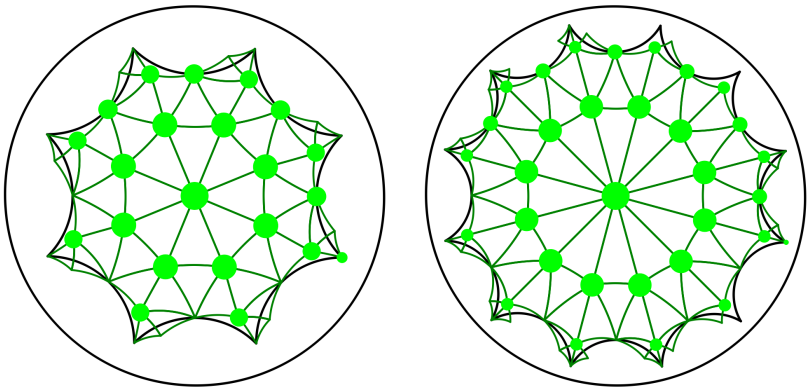

Figure V.19: Delaunay triangulations of $\mathbb{M}_2$ (left) and $\mathbb{M}_3$ (right) with dummy points generated with the sequential strategy with symmetries.

Let $\mathcal{P}$ be a set of input points in $\mathscr{D}_g$, and let $\mathcal{Q}_g$ be a set of dummy points for $\mathbb{M}_g$ generated with one of the sequential strategies described in Sections V.2.3 and V.2.4. We have developed a prototype code for the computation of $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$ with $\mathcal{S} = \mathcal{P} \bigcup \mathcal{Q}_g$. For genus 2 and 3, we have been able to experimentally obtain sets of dummy points with both sequential strategies with a precision of $g \times 512$ bits.[21] Following the initialization process for dummy points generated with one of the sequential strategies described in Section V.3.1, we have been able to construct initial Delaunay triangulations of $\mathbb{M}_2$ and

[21] Recall from Sections V.2.3 and V.2.4 that for genus higher than 3 the generation of dummy points fails due to insufficient accuracy even with $2048 \times g$ bits.

$\mathbb{M}_3$. We have also successfully located points in the triangulation, inserted new points, and removed existing vertices. For every operation, we verify that the combinatorial structure of the resulting triangulation is correct.

The initial triangulations of $\mathbb{M}_2$ and $\mathbb{M}_3$ defined by dummy points resulting from the sequential strategy are shown in Figure V.18. For dummy points resulting from the sequential strategy with symmetries, Figure V.19 shows the initial triangulations of $\mathbb{M}_2$ and $\mathbb{M}_3$. Figure V.20 shows the result of inserting points in the triangulations $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{Q}_2)$ and $\mathrm{DT}_{\mathbb{M}_3}(\mathcal{Q}_3)$ initialized with dummy points generated with the sequential strategy with symmetries. Note that, to produce Figure V.20, the code executes location and insertion of new points, as well as removal of unnecessary dummy points.

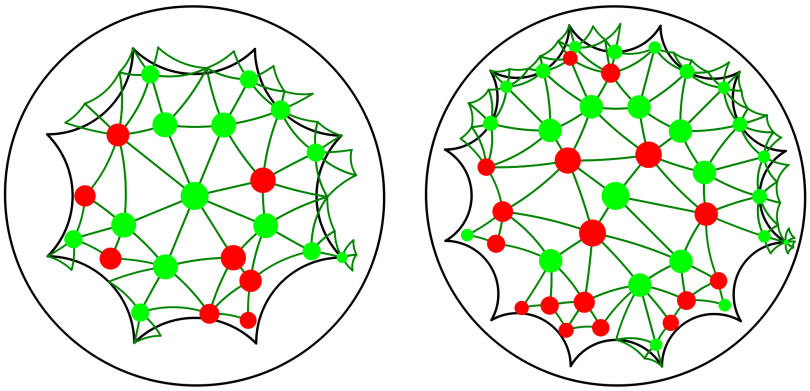

Figure V.20: Insertion of user-defined points (shown in red) in the Delaunay triangulations of $\mathbb{M}_2$ (left) and $\mathbb{M}_3$ (right) defined by dummy points generated with the sequential strategy with symmetries.

V.6 Conclusion

In this chapter we discussed an incremental algorithm that generalizes the algorithm for the Bolza surface discussed in Chapter III to surfaces of higher genus. We gave a data structure to represent the triangulation, detailed the construction and validity of sets of dummy points, and discussed the construction of the triangulation defined by the dummy points. We proposed an exact representation for the matrices of the generators of the group Γ_g and considered the practical difficulties that come with this representation. Finally, we presented a preliminary implementation of the proposed algorithm.

Topics that remain unexplored are the algebraic degree of the predicates needed for the construction of Delaunay triangulations of $\mathbb{M}_g$ for $g > 2$, and a representation of the matrices of the generators of Γ_g that enables efficient and exact computations.

Chapter VI

Conclusions and future work

In this thesis we studied ways to compute in practice Delaunay triangulations of symmetric hyperbolic surfaces. Such triangulations can be seen as periodic Delaunay triangulations of the hyperbolic plane.

We started by discussing an algorithm for the construction of Delaunay triangulations of the Bolza surface $\mathbb{M}_2$, the most symmetric hyperbolic surface of genus 2. We explored properties of the triangulation of $\mathbb{M}_2$ defined by a set of points $\mathcal{S}$ containing the set of dummy points $\mathcal{Q}_2$ proposed by Bogdanov *et al.* [BTV16]. We gave a representation of $\mathrm{DT}_{\mathbb{M}_2}(\mathcal{S})$ via uniquely defined images of its vertices and faces in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_2\mathcal{S})$. We discussed in detail the construction of an initial triangulation of $\mathbb{M}_2$ defined by the set $\mathcal{Q}_2$, as well as the insertion of new points in the triangulation. Other operations we examined were point location and vertex removal, providing a fully dynamic algorithm. We studied the algebraic degree of the predicates involved in the construction of Delaunay triangulations of $\mathbb{M}_2$.

Next, we presented an implementation of our algorithm in CGAL. We gave a brief overview of the CGAL library and discussed the existing framework, triangulation objects, and related concepts and models. We then presented the new objects that we introduce and discussed their integration in the library. We gave a detailed report on the resulting code and demo application, and presented experimental results to compare the effectiveness of our code with existing CGAL implementations.

We continued with a discussion on the formulation and implementation of an algorithm for the computation of Delaunay triangulations of symmetric hyperbolic surfaces $\mathbb{M}_g$ of genus higher than 2. To propose this extension, we studied methods for constructing sets of dummy points $\mathcal{Q}_g$ and presented the properties of each method. Similarly to the Bolza surface, we explored properties of triangulations of $\mathbb{M}_g$ defined by sets of points $\mathcal{S}$ containing $\mathcal{Q}_g$, which allowed us to represent $\mathrm{DT}_{\mathbb{M}_g}(\mathcal{S})$ via images of its vertices and faces in $\mathrm{DT}_{\mathbb{H}^2}(\Gamma_g\mathcal{S})$. We discussed the difficulties of implementing the proposed algorithm in practice, and gave experimental results from a preliminary implementation.

An immediate extension of our research is the proposal of exact and efficient predi-

cates for the computation of Delaunay triangulations of symmetric hyperbolic surfaces. It is also of practical interest to implement periodic meshes in the hyperbolic plane, for which this work is the basis.

On a more general note, an interesting representation of general closed orientable surfaces of genus 2 has been proposed by Aigon-Dupuy *et al.* [ADBC+05]. The fundamental domain of a general surface of genus 2 is represented as a non-regular octagon in $\mathbb{H}^2$ with angle sum 2π. One of the difficulties of the octagonal representation is that the fundamental domain is not a Dirichlet region, so many of the properties that we use in this thesis would not be valid. A construction of *pants decomposition* of the surface from this representation is described as well, in which a surface is represented as the union of *pairs of pants* [Bus92, Definition 3.1.1]. These two representations of hyperbolic surfaces could open the road to new algorithms for the computation of Delaunay triangulation of compact hyperbolic surfaces. The pants decomposition is a very handy representation, as it is adapted to general surfaces of genus higher than 1. Pants decomposition could be used for the computation of *local systoles* (i.e., the length of shortest non-contractible loops in each pair of pants) as opposed to the systole of the surface, leading to new algorithms for the computation of Delaunay triangulations of general hyperbolic surfaces.

Another very interesting direction is the computation of Delaunay triangulations of *Triply Periodic Minimal Surfaces* (TPMS). Such surfaces have locally minimal area and are embeddable in $\mathbb{R}^3$, where they are periodic in 3 directions. These surfaces also have negative Gaussian curvature, so they are hyperbolic and their universal cover is $\mathbb{H}^2$. In fact, all connected TPMS have genus at least 3 [Mee77]. Three of these surfaces, the Schwarz primitive, the Schwarz diamond, and the gyroid, can be covered by reflected copies of the hyperbolic triangle with angles $\frac{\pi}{2}$, $\frac{\pi}{4}$, and $\frac{\pi}{6}$. See Figure VI.1. Such surfaces find applications in biology and material sciences, as well as in architecture.

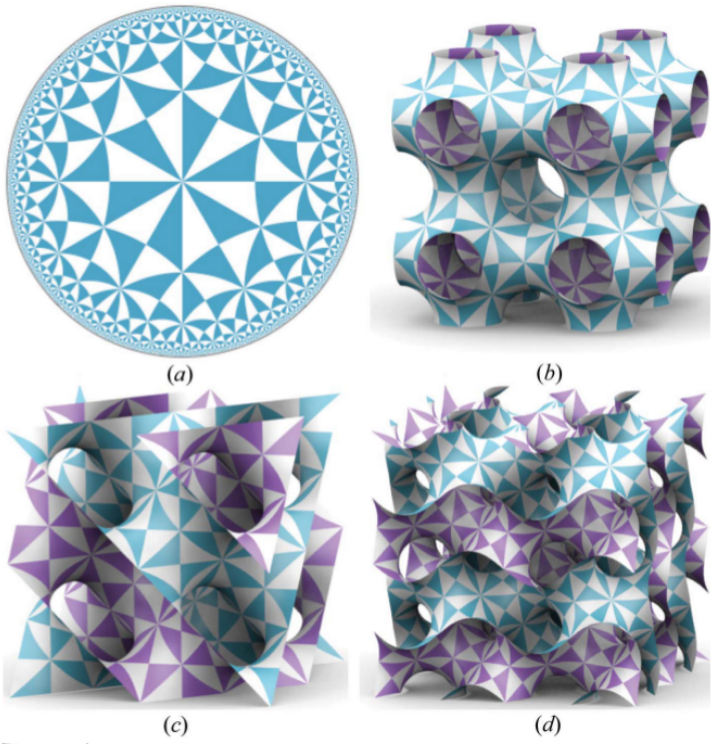

*Figure VI.1: Triply periodic minimal surfaces. Images taken from [ERH13a]. **(a):** Tiling of $\mathbb{H}^2$ with the hyperbolic triangle with angles $\frac{\pi}{2}, \frac{\pi}{4}$, and $\frac{\pi}{6}$. **(b):** The Schwarz primitive surface. **(c):** The Schwarz diamond surface. **(d):** The gyroid surface.*

Bibliography

[ADBC+05] Aline Aigon-Dupuy, Peter Buser, Michel Cibils, Alfred F Künzle, and Frank Steiner. Hyperbolic octagons and Teichmüller space in genus 2. *Journal of Mathematical Physics*, 46(3):33513, 2005. doi:10.1063/1.1850177.

[AGSS89] Alok Aggarwal, Leonidas J Guibas, James Saxe, and Peter W Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete & Computational Geometry*, 4(6):591–604, 1989.

[Arm83] Mark Anthony Armstrong. *Basic topology*. Springer Science & Business Media, 1983. doi:10.1007/978-1-4757-1793-8.

[ASG06] Elsa Abbena, Simon Salamon, and Alfred Gray. *Modern differential geometry of curves and surfaces with Mathematica*. Chapman and Hall/CRC, 3rd edition, 2006.

[AW43] Carl B Allendoerfer and André Weil. The Gauss-Bonnet theorem for Riemannian polyhedra. *Transactions of the American Mathematical Society*, 53(1):101–129, 1943. doi:10.1090/S0002-9947-1943-0007627-9.

[Bav92] Christophe Bavard. La systole des surfaces hyperelliptiques. *Prepubl. Ec. Norm. Sup. Lyon*, 71, 1992.

[BDT14] Mikhail Bogdanov, Olivier Devillers, and Monique Teillaud. Hyperbolic Delaunay complexes and Voronoi diagrams made practical. *Journal of Computational Geometry*, 5(1):56–85, 2014. URL: https://hal.inria.fr/hal-00961390, doi:10.20382/jocg.v5i1a4.

[Bea83] Alan F. Beardon. *The geometry of discrete groups*. Springer-Verlag New York, 1 edition, 1983. doi:10.1007/978-1-4612-1146-4.

[BFM+01] Christoph Burnikel, Stefan Funke, Kurt Mehlhorn, Stefan Schirra, and Susanne Schmitt. A separation bound for real algebraic expressions. In Friedhelm Meyer auf der Heide, editor, *Algorithms — ESA 2001*, pages

254–265, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. doi:10.1007/3-540-44676-1_21.

[BIT] Mikhail Bogdanov, Iordan Iordanov, and Monique Teillaud. 2D Hyperbolic Triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board. To appear.

[BJLT12] Vincent Borrelli, Saïd Jabrane, Francis Lazarus, and Boris Thibert. Flat tori in three-dimensional space and convex integration. *Proceedings of the National Academy of Sciences*, 2012. doi:10.1073/pnas.1118478109.

[Bow81] A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981. doi:10.1093/comjnl/24.2.162.

[BTV16] Mikhail Bogdanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations on orientable surfaces of low genus. In *Proceedings of the Thirty-second International Symposium on Computational Geometry*, pages 20:1–20:15, 2016. URL: https://hal.inria.fr/hal-01276386, doi:10.4230/LIPIcs.SoCG.2016.20.

[Bus92] P. Buser. *Geometry and spectra of compact Riemann surfaces*. Progress in Mathematics Series. Birkhäuser, 1992.

[BV86] N.L. Balazs and A. Voros. Chaos on the pseudosphere. *Physics Reports*, 143(3):109 – 240, 1986. doi:10.1016/0370-1573(86)90159-6.

[Car10] Manuel Caroli. *Triangulating point sets in orbit spaces*. Ph.D. Thesis, Université Nice Sophia Antipolis, December 2010. URL: https://tel.archives-ouvertes.fr/tel-00552215.

[CFF11] P. Chossat, G. Faye, and O. Faugeras. Bifurcation of hyperbolic planforms. *Journal of Nonlinear Science*, 21:465–498, 2011. URL: http://link.springer.com/article/10.1007%2Fs00332-010-9089-3, doi:10.1007/s00332-010-9089-3.

[cga] CGAL, Computational Geometry Algorithms Library. URL: http://www.cgal.org.

[Che90] L. Paul Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dartmouth College, Computer Science, Hanover, NH, 1990. URL: http://www.cs.dartmouth.edu/reports/TR90-147.pdf.

[CT09] Manuel Caroli and Monique Teillaud. 3D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.5 (and

further) edition, 2009. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic3Triangulation3Summary.

[CT16] 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.

[Dan] Al Danial. Count Lines of Code (cloc). URL: https://github.com/AlDanial/cloc.

[dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 1997.

[Deh12] M. Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Mathematische Annalen*, 72(3):413–421, 1912. doi:10.1007/BF01456725.

[Dev02] Olivier Devillers. On deletion in Delaunay triangulations. *International Journal of Computational Geometry & Applications*, 12(03):193–205, 2002.

[DH97] Nikolai P. Dolbilin and Daniel H. Huson. Periodic Delone tilings. *Periodica Mathematica Hungarica*, 34:1-2:57–64, 1997.

[DHJ15] Olivier Devillers, Samuel Hornus, and Clément Jamin. dD Triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.6 edition, 2015. URL: https://doc.cgal.org/latest/Triangulation/index.html.

[DPT02] Olivier Devillers, Sylvain Pion, and Monique Teillaud. Walking in a triangulation. *International Journal of Foundations of Computer Science*, 13:181–199, 2002. URL: https://hal.inria.fr/inria-00102194.

[DSY07] Zilin Du, Vikram Sharma, and Chee K Yap. Amortized bound for root isolation via Sturm sequences. In *Symbolic-Numeric Computation*, pages 113–129. Springer, 2007.

[DT03] Olivier Devillers and Monique Teillaud. Perturbations and vertex removal in a 3D Delaunay triangulation. In *14th ACM-Siam Symposium on Discrete Algorithms (SODA)*, pages 313–319, Baltimore, MA, United States, 2003. URL: https://hal.inria.fr/inria-00166710.

[DT11] 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.archives-ouvertes.fr/inria-00560388/, doi:10.1016/j.comgeo.2010.09.010.

[Dwy87] Rex A Dwyer. A faster divide-and-conquer algorithm for constructing Delaunay triangulations. *Algorithmica*, 2(1-4):137–151, 1987.

[Ebb17] Matthijs Ebbens. Delaunay triangulations on hyperbolic surfaces. Master's thesis, University of Groningen, The Netherlands, 2017. URL: http://fse.studenttheses.ub.rug.nl/id/eprint/15727.

[EITV] Matthijs Ebbens, Iordan Iordanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations of symmetric hyperbolic surfaces. In preparation.

[EITV18a] Matthijs Ebbens, Iordan Iordanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations of regular hyperbolic surfaces. In *9th International Conference on Curves and Surfaces*, volume 20, pages 1 – 20, Arcachon, France, Jun 2018. URL: https://hal.inria.fr/hal-01801136.

[EITV18b] Matthijs Ebbens, Iordan Iordanov, Monique Teillaud, and Gert Vegter. Systole of regular hyperbolic surfaces with an application to Delaunay triangulations. In *9th International Conference on Curves and Surfaces*, Arcachon, France, Jun 2018. URL: https://hal.inria.fr/hal-01803443.

[ERH13a] Myfanwy E. Evans, Vanessa Robins, and Stephen T. Hyde. Periodic entanglement I: networks from hyperbolic reticulations. *Acta Crystallographica Section A*, 69(3):241–261, May 2013. doi:10.1107/S0108767313001670.

[ERH13b] Myfanwy E. Evans, Vanessa Robins, and Stephen T. Hyde. Periodic entanglement II: weavings from hyperbolic line patterns. *Acta Crystallographica Section A*, 69(3):262–275, May 2013. doi:10.1107/S0108767313001682.

[Euc] Euclid. Στοιχεῖα Εὐκλείδου (Euclid's Elements). Compiled in HTML format by D. Mourmouras. URL: http://www.physics.ntua.gr/~mourmouras/euclid.

[EV] Matthijs Ebbens and Gert Vegter. Personal communication.

[FK92] Hershel M. Farkas and Irwin Kra. Riemann surfaces. In *Graduate texts in Mathematics*, volume 17. Springer, 1992. doi:10.1007/978-1-4612-2034-3.

[For87] Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2(1-4):153, 1987.

[GAP16] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.8.6*, 2016. URL: http://www.gap-system.org.

[Gre60] Martin Greendlinger. Dehn's algorithm for the word problem. *Communications on Pure and Applied Mathematics*, 13(1):67–83, 1960. doi:10.1002/cpa.3160130108.

[GS85] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM transactions on graphics (TOG)*, 4(2):74–123, 1985.

[Hil01] David Hilbert. Ueber Flächen von constanter Gaussscher Krümmung. *Transactions of the American Mathematical Society*, 2, 1901. doi:10.1090/S0002-9947-1901-1500557-5.

[HN59] Philip Hartman and Louis Nirenberg. On spherical image maps whose Jacobians do not change sign. *American Journal of Mathematics*, 81(4):901–920, 1959. URL: http://www.jstor.org/stable/2372995.

[IT] Iordan Iordanov and Monique Teillaud. 2D Periodic Hyperbolic Triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board. To appear.

[IT17] Iordan Iordanov and Monique Teillaud. Implementing Delaunay triangulations of the Bolza surface. In *Proceedings of the Thirty-third International Symposium on Computational Geometry*, pages 44:1–44:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.44.

[JPT14] Clément Jamin, Sylvain Pion, and Monique Teillaud. 3d triangulation data structure. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 4.5 edition, 2014. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgTDS3Summary.

[JR80] M. Jungerman and G. Ringel. Minimal triangulations on orientable surfaces. *Acta Math.*, 145:121–154, 1980. doi:10.1007/BF02414187.

[Kat92] S. Katok. *Fuchsian Groups*. Chicago Lectures in Mathematics. University of Chicago Press, 1992.

[Kru13] Nico Kruithof. 2D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 (and further) edition, 2013. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic2Triangulation2Summary.

[Law77] Charles L Lawson. Software for C^1 surface interpolation. In *Mathematical software*, pages 161–194. Elsevier, 1977.

[Lee00] John M. Lee. *Introduction to Topological Manifolds*. Springer-Verlag, New York, 2000.

[mag] The Magma Development Team. *Magma Computational Algebra System*. URL: http://magma.maths.usyd.edu.au/magma/.

[Mee77] William Meeks. The conformal structure and geometry of triply periodic minimal surfaces in $\mathbb{R}^3$. *Bulletin of The American Mathematical Society*, 83, 1977. doi:10.1090/S0002-9904-1977-14218-3.

[OR03] John Joseph O'Connor and Edmund Frederick Robertson. Word problems for groups. The MacTutor History of Mathematics archive, 2003. URL: http://www-history.mcs.st-andrews.ac.uk/HistTopics/Word_problems.html.

[Rat06] J. Ratcliffe. *Foundations of Hyperbolic Manifolds*, volume 149 of *Graduate Texts in Mathematics*. Springer, 2006. doi:10.1007/978-0-387-47322-2.

[Rup95] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548 – 585, 1995. doi:https://doi.org/10.1006/jagm.1995.1021.

[STV08] François Sausset, Gilles Tarjus, and Pascal Viot. Tuning the fragility of a glassforming liquid by curving space. *Physical Review Letters*, 101:155701(1)–155701(4), 2008. doi:10.1103/PhysRevLett.101.155701.

[Ung05] A.A. Ungar. Einstein's special relativity: Unleashing the power of its hyperbolic geometry. *Computers & Mathematics with Applications*, 49(2):187 – 221, 2005. doi:https://doi.org/10.1016/j.camwa.2004.10.030.

[Wat81] D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981. doi:10.1093/comjnl/24.2.167.

[YD95] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific, Singapore, 2nd edition, 1995. doi:10.1142/9789812831699_0011.

[Yvi97] Mariette Yvinec. 2D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 0.9 edition, 1997. URL: https://doc.cgal.org/latest/Triangulation_2/index.html.