A computer-based algorithmic approach to the study of automorphism groups, orbits of smooth rational curves, unirationality & projective models of K3 surfaces

Cedric Mazet

Aix*Marseille université
Socialement engagée

NNT/NL : 2022AIXM0226/008ED184

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 30 juin 2022 par

Cédric MAZET

A computer-based algorithmic approach to the study of automorphism groups, orbits of smooth rational curves, unirationality and projective models of K3 surfaces

Discipline
Mathématiques

Spécialité
Mathématiques fondamentales

École doctorale
ED 184 Mathématiques et Informatique

Laboratoire
Institut de Mathématiques de Marseille

Composition du jury

Alice GARBAGNATI Università Statale di Milano	Rapporteuse
Davide VENIANI Stuttgart University	Rapporteur
Alessandra SARTI Université de Poitiers	Examinatrice
Pierre LAIREZ École Polytechnique	Examinateur
Frédéric MANGOLTE Université d'Angers	Président du jury
Xavier ROULLEAU Aix-Marseille Université	Directeur de thèse

Affidavit

I, undersigned, Cédric MAZET, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of Professor Xavier ROULLEAU, in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the French national charter for Research Integrity and the Aix-Marseille University charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Marseille, May 31, 2022.

Affidavit

Je soussigné, Cédric MAZET, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique du Professeur Xavier ROULLEAU, dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat.

Ce travail n'a pas été précédemment soumis en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Fait à Marseille le 31 Mai 2022.

Abstract

The initial aim of this thesis consisted in determining automorphism groups and upper bounds on the number of orbits of smooth rational curves on surfaces in the family of $K3$ surfaces having a Néron-Severi group isomorphic to the lattice with Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix} \qquad \text{with } 1 \leq t \leq 50$$

with respect to a fixed basis. To this end, we put computer science at the service of pure mathematics and implemented various computer-based algorithmic solutions that take advantage of a wide array of tools and modern techniques. These solutions not only enabled us to perform a complete study of the family of $K3$ surfaces mentioned above by determining projective models, computing automorphism groups, studying the orbits of smooth rational curves, and discussing the unirationality of their moduli spaces, hence enabling us to provide results far exceeding the objectives which had been set for this thesis, but also turn out to have a framework of application which goes far beyond the family of surfaces mentioned earlier. From the outset of this thesis, we indeed had in mind to develop solutions with a broad scope of application. This endeavor resulted in the production of many computer-based solutions for the study of $K3$ surfaces that will hopefully open up new perspectives and help popularize even more the field of study of $K3$ surfaces. Please note that all programs produced during this thesis are released in public access: All computer-based solutions produced during this thesis are detailed and available for download on K3surfaces.com.

Keywords: K3 Surfaces, Pure Mathematics, Computer Science, Python, Sage, Scipy, Multiprocessing, Pool, Automorphisms, Rational curves, Algebraic Geometry, Projective models, Parallelism, Borcherds' method, K3surfaces.com

Résumé

Les objectifs initialement fixés pour cette thèse consistaient à déterminer les groupes d'automorphismes ainsi que des bornes supérieures sur le nombre d'orbites de courbes rationnelles sur les surfaces $K3$ appartenant à la famille des surfaces ayant un groupe de Néron-Severi isomorphe au réseau entier avec matrice de Gram

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix} \qquad \text{avec } 1 \leq t \leq 50$$

par rapport à une base fixée. Nous avons pour cela mis l'outil informatique au service des mathématiques fondamentales en implémentant des solutions algorithmiques tirant parti d'outils modernes et variés. Les programmes qui ont découlé de cette démarche nous ont non seulement permis de mener une étude complète de ces surfaces en calculant explicitement leurs automorphismes, orbites de (-2)-courbes sous l'action de ces derniers, modèles projectifs, unirationalité des espaces des modules, dépassant ainsi largement notre objectif initial d'étude, mais ont aussi un champ d'application allant bien au-delà de ces surfaces. Depuis le début de cette thèse, nous avons en effet été motivés par la volonté de toujours dépasser les cas particuliers et spécificités afin de produire des solutions ayant une portée généraliste assumée. Notre entreprise a ainsi résulté en la production de nombreuses solutions mettant l'outil informatique au service de la géométrie algébrique et des surfaces $K3$ qui, nous l'espérons, ouvriront de nouvelles perspectives d'étude pour ces dernières. Nous tenons à mentionner que tous les programmes réalisés pendant cette thèse sont accessibles via K3surfaces.com et que leur utilisation y est expliquée en détails.

Mots-clés : Surfaces $K3$, Maths pures, Informatique, Python, Sage, Scipy, Multiprocessing, Pool, Automorphismes, Courbes rationnelles, Géométrie Algébrique, modèles projectifs, méthode de Borcherds, K3surfaces.com

Acknowledgments

None of this would have been possible without Professor Xavier Roulleau. I was fortunate to be able to benefit from his ideas, expertise and knowledge which enabled me to overcome any obstacle. He invested a considerable amount of time on me, much more than a student should expect from an advisor. I deeply thank Professor Roulleau for his support during my time as an MSc & PhD student.

I want to express my profound gratitude to the reviewers of my thesis, Professor Alice Garbagnati & Doctor Davide Cesare Veniani, for the time spent reviewing my thesis and their positive feedback.

I want to thank my longtime friend Théo Petropoulos for his encouragement and advice as an expert web developer and cyber security professional regarding the online aspects of this thesis.

More than anything, I want to thank from the bottom of my heart my mother, my father and my sister for their love and support.

Introduction

Denote by X an algebraic $K3$ surface over the field of complex numbers.

Two classical results were established by Sterk in his article [20, Theorem 0.1] *Finiteness results for algebraic K3 surfaces*:

- $\mathrm{Aut}(X)$ is a finitely generated group.
- The number of orbits of (-2)-curves under the action of $\mathrm{Aut}(X)$ is finite.

These results enabled our advisor to throw at us the main challenge to be accomplished in order to achieve this doctoral project: For $1 \leq t \leq 50$, we had to determine a generating set of the automorphism group of the $K3$ surface X_t with Néron-Severi group isomorphic to the integral lattice with Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

with respect to a fixed basis. We were also tasked with finding an upper bound on the number of orbits of smooth rational curves on each such surface by using the acquired knowledge of their respective automorphism groups to our advantage. It is worth mentioning that Xavier Roulleau provided us with constant support, many ideas and gave us leeway in terms of the approaches and techniques to be used in order to reach the goals he had set for this thesis. We made the most of this opportunity by using an innovative computer-based algorithmic approach to the study of $K3$ surfaces. As will be shown in this dissertation, the solutions developed and implemented during this thesis have a reach that goes far beyond the scope of the above-mentioned family of $K3$ surfaces X_t. The content available on K3surfaces.com bears witness to this fact. We dwell on this in more detail in the introduction to Part I of this thesis.

Our computer-based algorithmic approach opens news doors not only for the study of automorphism groups and orbits of smooth rational curves on complex $K3$ surfaces, but also for the study of their projective models. Our computer-based algorithmic approach enables us to offer a new perspective on a classical result due to Saint-Donat & Morrison which is known to provide a precise description of the role of ample classes regarding embeddings of $K3$ surfaces into projective spaces. Our approach also takes advantage of the fact that Xavier Roulleau released a Magma implementation of a quite special algorithm along with the publication of his 2019 article [15]. Let X be a $K3$ surface. Roulleau's program takes as input a Gram matrix of the Néron-Severi group $S = \mathrm{NS}(X)$, an ample class P_0, integers d and u_b, to output the set

$$\{C \in \mathrm{NS}(X) \mid \langle C, C\rangle_S = d,\ \langle C, P_0\rangle \leq u_b\}$$

of classes C of divisors of self-intersection $\langle C, C\rangle_S = d$ having an intersection product with P_0 less than or equal to u_b. When $d = -2$, Prof. Roulleau's program is capable of identifying classes of smooth rational curves $C \simeq \mathbb{P}^1$ among the classes of self-intersection -2. This tool provides a gateway to knowledge of concrete data on classes of curves having a prescribed self-intersection, and more specifically on classes of smooth rational curves, which are known to play a central role on K3 surfaces. We thus made use of Prof. Roulleau's program to produce a large database of classes of not only smooth rational curves, but also of classes having any prescribed self-intersection on the surfaces we were tasked to study. Taking advantage of this mass-produced data, we pushed onto the path devised by Roulleau in [16, 15] and used a computer-based algorithmic approach to implement Roulleau's methods on an industrial scale. This endeavor resulted in the production of our **proj_mod** suite which offers tools such as **CGS**, **PModChecker** or an universal ampleness tester **AmpTester**. These solutions will hopefully open doors to other researchers and encourage them to take up the torch on the computer-based study of $K3$ surfaces. We did our best to ensure that this thesis can be used as a sound, safe and accessible ground for others to obtain even further developments in the future.

Introduction to Part I of this thesis

Denote by X_t a complex $K3$ surface with Néron-Severi group isomorphic to the integral lattice with Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

where the parameter t is assumed to be a positive integer. As mentioned earlier, we were tasked with the study of these surfaces for $1 \leq t \leq 50$. We now introduce the various solutions implemented during this thesis in order to deal with the challenges of explicitly computing the automorphism group and determining the orbits of smooth rational curves under its action on each of these surfaces. It turns out that the reach of these solutions goes well beyond the scope of these surfaces and gives a very general scope of application to the content of this thesis. We start by presenting the context in which our work fits as a development. Fields medalist Richard E. Borcherds introduced a method to compute the generators of the automorphism group of Lorentzian lattices with possible applications to $K3$ surfaces in two articles [1] and [2] published in the late eighties and early nineties. Borcherds' method was then applied for the first time to $K3$ surfaces in 1998 with Shigeyuki Kondō's groundbreaking article [8]. From this moment and until the end of the first decade of our century, mathematicians such as Ujikawa, Dolgachev, Keum, and Kondō again ([7, 6, 4, 10]) made use of Borcherds' method to compute automorphism groups of various $K3$ surfaces. In 2013, Professor Ichiro Shimada gave a new life to Richard E. Borcherds' three-decades-old material in his article *An algorithm to compute the automorphism groups of K3 surfaces* [19]. Carried out as part of a publication grant entitled Computational study of K3 surfaces $(2013/2016)$ and followed by another grant, this time entitled Computational study of algebraic geometry, Shimada's article [19] is unquestionably a massive step toward the full automation of the computation of generators of the automorphism groups of $K3$ surfaces. Professor

Ichiro Shimada, founding father of the computer-based algorithmic approach to the study of $K3$ surfaces, thus provided a sound theoretical background and outlined many of the essential procedures and building blocks required in order to carry out Borcherds' method as an algorithmic method. However, neither a functional program nor a single line of code was released since the publication of Prof. Shimada's article. Almost a decade has passed since Shimada's article, and no significant progress on the subject has been made. As mentioned by Giacomo Mezzedimi in his PhD thesis, defended in 2021,

> "*Shimada presents an algorithm to compute the automorphism group of these $K3$ surfaces; however the full automorphism group can only be computed for a finite number of Picard lattices [...]*
>
> *When the automorphism group becomes infinite, very little is known. For example, we can describe the full automorphism group only of some $K3$ surfaces.*"
>
> Giacomo Mezzedimi [12], PhD thesis, October 2021.

Indeed, Shimada's article was not generalist and focused on examples without explicitly highlighting a general application framework for Borcherds' method. In addition, many grey areas surrounded the steps that have to be taken to implement essential procedures described in Shimada's article. Various fundamental aspects essential to the generalization, optimization and implementation of the processes were often ignored or treated in a minimalist way. Shimada's article [19] was not intended to be a manual for the implementation of the various procedures that can be found therein. As a result, many challenges had to be overcome. First, we had to familiarize ourselves with Shimada's super fast-paced style to make the best possible use of the invaluable information contained in his article [19]. Moreover, many procedures from this article involve material from another article [18] due to Shimada, which therefore also had to be mastered. We then had to determine whether a general and precisely defined framework of application for Borcherds' method could be devised from Shimada's work. The answer is positive: It is indeed possible to do so. We, therefore, had to identify

the holes to be filled and missing pieces in order to bring to life and fully automate all the material which can be found in Shimada's article. These holes and missing pieces seem to have obstructed the path to a generalized implementation of the method for almost ten years. At the time we write these lines, we cannot find any trace of an implementation of Shimada's material on the internet that could rival what has been produced during this thesis. To be precise, there is nothing. Despite Shimada's article [19] being published almost a decade ago, i.e., in 2013, no sign of an elementary, limited or even partial implementation can be found. When released in 2022, our thesis put an end to this situation. Going back to our story, we have to mention that the first stage of our endeavor was carried out while having in mind our goal of producing a generalized implementation of Borcherds' method. That is, an implementation whose scope of application goes much beyond a handful of particular cases. Our desire for generality drove us to identify explicit conditions of applicability for Borcherds' method from the sound foundations laid by Shimada in 2013, and naturally led us to design and implement automated procedures enabling us to test whether a given $K3$ complex algebraic surface satisfies these conditions. We then had to move on to the implementation of the method itself. For versatility and flexibility purposes, our language of choice was naturally Python. We extensively used the Sage library, which includes many advanced mathematical features. This library was so convenient for us that we worked most of the time within a Sage / Python $3.8.5$ environment through a Sage terminal. We have been careful to produce flexible and accessible solutions requiring only a bare minimum of input data to be executed. Furthermore, our programs provide complete automation. For instance, no matter if it is to set up the ambient conditions required to execute the method, test whether Borcherds' method can be applied, or execute the method itself, everything is performed automatically. We also did our best to ensure that Borcherds' method can benefit from every ounce of computational power available on the machine on which it is executed. Indeed, we live in an era during which most machines take advantage of parallel processing. What would be the point of making daily use of expensive pieces of hardware to not even

use the full extent of their processing power for mathematical research? We, therefore, redesigned all our solutions with parallel computing in mind. Having used Python from the start enabled us to make a smooth transition to the use of process-based parallelism, thus enabling us to make the best possible use of the processing power of the central processing units on our machines by deploying various internal procedures of Borcherds' method in parallel. In particular, we fully took advantage of the **Pool** object from the Python **multiprocessing** library. This object, as indicated in the official Python documentation, offers a convenient means of parallelizing the execution of a function across multiple input values. Doing so enabled us to produce a modernized version of Borcherds' method: The **Poolized** Borcherds' method. Through the use of the Pool object, the burden of executing various computationally intensive procedures which are part of Borcherds' method is distributed over various worker processes in such a way as to take advantage of the multi-core architecture of modern CPUs. Doing so thus leads to a significant decrease in computation times. We were still hungry for challenge and wanted to push our enterprise of parallelizing Borcherds' method even further. This aspiration naturally led us to take a step forward toward parallelizing the Borcherds' method at a broader scale. To this end, we implemented solutions to parallelize the exploration of the chamber structure and the computation of the sets of walls of chambers. This approach, detailed both online and in the section 1.11 of this document, is a first step that will hopefully open many doors, broaden the perspectives regarding the parallel deployment of Borcherds' method and, more generally, enable this thesis to reinforce the **interface** between **pure mathematics** and **computer science**.

Before proceeding further, we have to mention that all the solutions presented in this manuscript exist as fully functional computer-based solutions.

There is nothing conceptual in our work: K3surfaces.com testifies to this fact.

We now introduce the subject matter covered in the first part of this document.

Let X be a complex $K3$ surface with Picard number $\rho_X < 20$ and Néron-Severi lattice $S = \mathrm{NS}(X)$ with Gram matrix G_S with respect to a fixed basis $\mathcal{B}_S$ for S. Denote by $\mathcal{P}_S$ the connected component of

$$\{x \in S \mid \langle x, x \rangle_S > 0\}$$

containing ample classes. We build on the solid foundations which have been laid by Shimada in [19] regarding Borcherds' method: The Néron-Severi group $S = \mathrm{NS}(X)$ of the complex algebraic $K3$ surface X under study should be embedded into a suitable even hyperbolic lattice $\mathbb{L}$ chosen according to the value of the Picard number of X, as indicated below:

Picard number ρ_X	Recommended ambient lattice
$1 \leq \rho_X < 10$	$U \oplus E_8(-1)$
$10 \leq \rho_X < 18$	$U \oplus E_8(-1) \oplus E_8(-1)$
$18 \leq \rho_X < 20$	$U \oplus E_8(-1) \oplus E_8(-1) \oplus E_8(-1)$

When possible, we recommend picking the ambient lattice $\mathbb{L}$ having the smallest possible rank among the three possible lattices displayed in this table. Indeed, choosing an ambient lattice of higher rank than what is recommended in the above table will decrease the performance of Borcherds' method. Before we go any further, we need to clarify some notational conventions. We will often write ρ instead of ρ_X. Let $N = \mathrm{rank}(\mathbb{L})$ and assume that a basis

$$\mathcal{B}_S = \{s_1, s_2, \ldots, s_\rho\}$$

for S and a basis

$$\mathcal{B}_{\mathbb{L}} = \{l_1, l_2, \ldots, l_N\}$$

for the lattice $\mathbb{L}$ are fixed. We use the notation

$$[\gamma_1, \gamma_2, \ldots, \gamma_\rho]_S$$

to denote the row vector of coordinates with respect to the basis $\mathcal{B}_S$ of the element

$$\gamma_1 s_1 + \gamma_2 s_2 + \cdots + \gamma_\rho s_\rho \in S.$$

Similarly, we denote by

$$[\beta_1, \beta_2, \ldots, \beta_N]_{\mathbb{L}}$$

the vector of coordinates with respect to $\mathcal{B}_{\mathbb{L}}$ of the element

$$\beta_1 l_1 + \beta_2 l_2 + \cdots + \beta_N l_N \in \mathbb{L}.$$

We assume that the Néron-Séveri group of the surface under study has been primitively embedded into a suitable ambient even hyperbolic lattice $\mathbb{L}$. That is, we assume known the data of elements

$$v_i = \sum_{j=1}^{N} \eta_j^{(i)} l_j$$

for $1 \leq i \leq N$ such that a mapping

$$\iota : S \hookrightarrow \mathbb{L}$$

embedding S primitively into $\mathbb{L}$ can be defined by

$$\iota : s_i \longmapsto v_i.$$

That is,

$$\iota : \gamma_1 s_1 + \gamma_2 s_2 + \cdots + \gamma_\rho s_\rho \in S \longmapsto \gamma_1 v_1 + \gamma_2 v_2 + \cdots + \gamma_\rho v_\rho \in \mathbb{L}$$

Note that in terms of coordinates vectors, this mapping is defined as

$$\iota : [\gamma_1, \gamma_2, \ldots, \gamma_\rho]_S \longmapsto \left[\sum_{j=1}^{N} \gamma_j \eta_1^{(j)}, \sum_{j=1}^{N} \gamma_j \eta_2^{(j)}, \ldots, \sum_{j=1}^{N} \gamma_j \eta_N^{(j)} \right].$$

The set

$$\{x \in \mathbb{L} \otimes \mathbb{R} \mid \langle x, x \rangle_{\mathbb{L}} > 0\}$$

has two connected components. The connected component containing $\iota(\mathcal{P}_S)$ is called the positive cone of $\mathbb{L}$ and denoted by $\mathcal{P}_{\mathbb{L}}$. A closed subset $D \subset \mathcal{P}_{\mathbb{L}}$ is called a chamber whenever it has non-empty interior and there exists a set

$$\Delta \subset \mathcal{N}_{\mathbb{L}} = \{x \in \mathbb{L} \otimes \mathbb{R} \mid \langle x, x \rangle_{\mathbb{L}} < 0\}$$

such that D can be expressed as

$$D = \{x \in \mathbb{L} \otimes \mathbb{R} \mid \forall v \in \Delta,\ \langle x, v \rangle_{\mathbb{L}} \geq 0\} \cap \mathcal{P}_{\mathbb{L}}.$$

We denote by $\overline{C}$ the topological closure of a set C. The collection

$$\mathcal{C}_{\mathbb{L}} = \left\{ \overline{C} \mid C \text{ is a connected component of } \mathcal{P}_{\mathbb{L}} \setminus \bigcup_{v \in \mathcal{F}} (v)^{\perp},\ \mathrm{Int}(\overline{C}) \neq \emptyset \right\}$$

is called a chamber structure on $\mathcal{P}_{\mathbb{L}}$, or a $\mathcal{P}_{\mathbb{L}}$-chamber structure. Chambers in $\mathcal{C}_{\mathbb{L}}$ will be referred to as $\mathcal{P}_{\mathbb{L}}$-chambers. Let

$$\mathcal{R}_{\mathbb{L}} = \{x \in \mathbb{L} \mid \langle x, x \rangle_{\mathbb{L}} = -2\}.$$

In practice, we take

$$\mathcal{F} = \mathcal{R}_{\mathbb{L}}$$

to define a chamber structure on $\mathcal{P}_{\mathbb{L}}$, where $\mathbb{L}$ is one of three lattices specified in the above-mentioned table. We will often use the notation $\mathcal{D}$ to denote a $\mathcal{P}_{\mathbb{L}}$-chamber. A fact that should be highlighted is that a chamber structure on $\mathcal{P}_{\mathbb{L}}$ induces a chamber structure on the positive cone $\mathcal{P}_S$ of the Néron-Severi group

S, the latter being assumed to be primitively embedded into $\mathbb{L}$. We show in section 1.2 that whenever a $\mathcal{P}_{\mathbb{L}}$-chamber structure $\mathcal{C}_{\mathbb{L}}$ is given, the collection

$$\mathcal{C}_S = \{\mathcal{D} \cap \mathcal{P}_S \mid \mathcal{D} \in \mathcal{C}_{\mathbb{L}}, \exists U \subset \mathcal{P}_S, U \neq \emptyset, U \text{ open s.t. } U \subset \mathcal{D} \cap \mathcal{P}_S\}$$

is a chamber structure on $\mathcal{P}_S$. Chambers $D \in \mathcal{C}_S$ are called $\mathcal{P}_S$-chambers. The intersection of a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ with $\mathcal{P}_S$ thus defines a $\mathcal{P}_S$-chamber whenever the resulting set has non-empty interior. A $\mathcal{P}_{\mathbb{L}}$-chamber which induces a $\mathcal{P}_S$-chamber is said to be $\iota(S)$-nondegenerate, or is said to possess the $\iota(S)$-nondegeneracy property. The $\iota(S)$-prefix is used to emphasize the fact that this property of a $\mathcal{P}_{\mathbb{L}}$-chamber depends on the embedding $\iota : S \hookrightarrow \mathbb{L}$ used to embed S into $\mathbb{L}$. A central notion that will be essential throughout our study is the notion of Weyl vector of a $\mathcal{P}_{\mathbb{L}}$-chamber. Each such chamber is indeed uniquely characterized by its Weyl vector. See definition 11 from section 1.1.2 for more details. Whenever a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ induces a $\mathcal{P}_S$-chamber $D = \mathcal{D} \cap \mathcal{P}_S$, the convention is that the induced $\mathcal{P}_S$-chamber D inherits the Weyl vector of the chamber $\mathcal{D}$. Another critical attribute of a $\mathcal{P}_S$-chamber D with Weyl vector w is its *set of walls*, denoted by $\Omega(D)$. We will see that this set can be obtained from the data of the Weyl vector of D. More generally, many of the computations and procedures involving a $\mathcal{P}_S$-chamber D involve its Weyl vector at one time or another. An important thing to point out before proceeding further is that the intersection $\text{Nef}(X) \cap \mathcal{P}_S$ of the Nef cone of X with the positive cone $\mathcal{P}_S$ is naturally tiled by chambers of the induced chamber structure $\mathcal{C}_S$. This natural chamber substructure covering $\text{Nef}(X) \cap \mathcal{P}_S$ is moreover cut by walls defined by the respective orthogonal complements in $\mathcal{P}_S$ of classes of smooth rational curves on X. Consider the $K3$ lattice

$$H^2(X, \mathbb{Z}) \simeq U^{\oplus 3} + E_8(-1)^{\oplus 2}$$

and denote by $\mathbf{H}$ the subgroup of transformations in $O^+(S)$ lifting to Hodge

isometries in $H^2(X,\mathbb{Z})$. Let

$$\begin{aligned}\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X)\cap\mathcal{P}_S) &= \{g\in\mathbf{H}\mid g \text{ preserves } \mathrm{Nef}(X)\cap\mathcal{P}_S\}\\ &\subset \mathbf{H}\subset O^+(S)\subset O(S)\end{aligned}$$

be the subgroup of transformations in $\mathbf{H}$ preserving $\mathrm{Nef}(X)\cap\mathcal{P}_S$. This group is a prominent character in regards to one of our main objects of study: Borcherds' method, whose purpose consists in producing a generating set of

$$\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X)\cap\mathcal{P}_S)$$

by exploring and processing the portion of the induced chamber structure on $\mathcal{P}_S$ covering $\mathrm{Nef}(X)\cap\mathcal{P}_S$. In section 1.7, we explain what the sentence *exploring and processing* the chamber structure means. In this introduction, specifying the bare minimum required for a good understanding of the method will be enough.

Exploring the portion of the $\mathcal{P}_S$-chamber structure over $\mathrm{Nef}(X)\cap\mathcal{P}_S$ requires the data of an initial $\mathcal{P}_S$-chamber D_0 contained in $\mathrm{Nef}(X)\cap\mathcal{P}_S$ to be used as a starting point to initiate the exploration. As indicated in Shimada's article [19], the classical theory fortunately always provides the Weyl vector w_0 of an initial chamber $\mathcal{D}_0$ of the $\mathcal{P}_{\mathbb{L}}$-chamber structure no matter which lattice $\mathbb{L}$ is chosen among the three lattices presented in the table introduced earlier.

$$\iota : S \hookrightarrow \mathbb{L}$$

There is, however, no guarantee that $\mathcal{D}_0$ will be $\iota(S)$-nondegenerate. Indeed, the $\iota(S)$-nondegeneracy property of $\mathcal{D}_0$ depends on the embedding ι used to embed the Néron-Severi group S of the $K3$ surface under study into $\mathbb{L}$. In his article [19], Shimada provides a criterion to determine whether a given $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ is nondegenerate. Our implementation of this criterion is the procedure **Degen-Test**, whose mechanics are explained in section 1.2. Using this criterion requires the input data of an ample class a_0 and of the Weyl vector of a chamber $\mathcal{D}_0$. The

mechanics behind this test take advantage of the fact that an ample class a_0 is by definition an element of $\mathrm{Nef}(X) \cap \mathcal{P}_S$ so that one can determine quite easily whether the image $\iota(a_0)$ of the ample class a_0 under the embedding $\iota : S \hookrightarrow \mathbb{L}$ belongs to the interior of $\mathcal{D}_0 \cap \mathcal{P}_S$. The intersection $\mathcal{D}_0 \cap \mathcal{P}_S$ has non-empty interior whenever this is the case, hence ensuring the $\iota(S)$-nondegeneracy of $\mathcal{D}_0$, and we obtain at the same time that

$$\mathcal{D}_0 \cap \mathcal{P}_S \subset \mathrm{Nef}(X) \cap \mathcal{P}_S.$$

In his article [19], Shimada also provides the outline of a procedure, which, in the framework of an embedding

$$\iota : S \hookrightarrow \mathbb{L}$$

and given the input data of a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ and of an ample class a_0 such that $\iota(a_0)$ does not belong to the interior of $\mathcal{D}_0 \cap \mathcal{P}_S$, may lead to an isometry $\tau : \mathbb{L} \longrightarrow \mathbb{L}$ which can be used to define an updated embedding

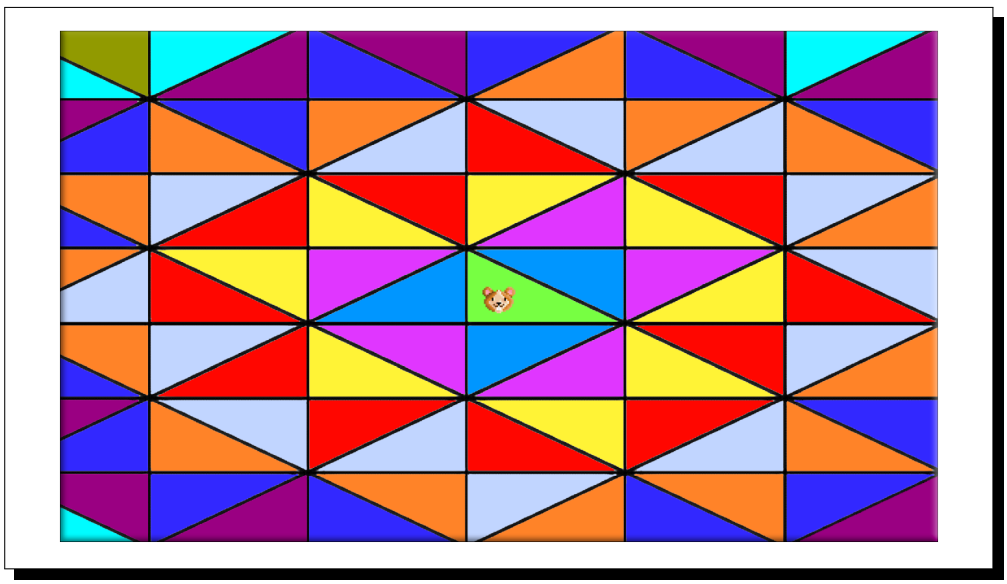
$$\tau \circ \iota : S \hookrightarrow \mathbb{L}$$

under which the chamber $\mathcal{D}_0$ possesses the property of $(\tau \circ \iota)(S)$-nondegeneracy. A favorable outcome to Shimada's procedure will indeed make the image of a_0 under the updated embedding satisfy

$$(\tau \circ \iota)(a_0) \in \mathrm{Int}(\mathcal{D}_0 \cap \mathcal{P}_S)$$

thus ensuring the $(\tau \circ \iota)(S)$-non-degeneracy of $\mathcal{D}_0$ and the fact that the chamber it induces is contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. We have to mention that Shimada's embedding update procedure outlined in [19, Section 8] has probably been for almost a decade one of the major obstructions to the production of a functional and generalized implementation of Borcherds' method. Building on Shimada's original procedure, we worked our way toward a modernized embedding update procedure, which, once implemented, brings many improvements compared to

our implementation of Shimada's original procedure. Going back to Borcherds' method, note that as soon as a suitable initial chamber $\mathcal{P}_S$-chamber contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is obtained, the exploration can begin. We start by focusing on explaining how Borcherds' method moves inside of the chamber structure. It is essential to have in mind the fact that chamber structure can be viewed as a tiling over $\mathrm{Nef}(X) \cap \mathcal{P}_S$, as illustrated in the following figure.



A fundamental concept related to the tiling of $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is the notion of level for chambers, which enables us to layer the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ with respect to a reference point. Fix an initial $\mathcal{P}_S$-chamber D_0. The notion of level for chambers is defined iteratively, as follows:

- The initial chamber D_0 is the only level 0 chamber.
- A chamber adjacent to a level $l-1$ chamber but not adjacent to a level $l-2$ chamber is said to be of level l.

The figure above depicts a genuine representation of a portion of the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$, where X is the $K3$ surface X_{42} in Picard 3 and

where an initial chamber, green-colored and located at the center of the picture, has been chosen as a reference point. In terms of level, the chamber structure in this picture can be described as follows:

- The chamber colored in green at the center is the initial chamber of level 0 used as a reference point for the chamber structure's layering. There is only one chamber of level 0.
- Chambers colored in clear blue are the chambers of level 1.
- Chambers colored in bright purple are the chambers of level 2.
- Chambers colored in yellow are the chambers of level 3.
- Chambers colored in red are the chambers of level 4.
- Chambers colored in a grey / blueish color are the chambers of level 5.
- Chambers colored in orange are the chambers of level 6.

We already mentioned that Borcherds' method is an iterative procedure during which a portion of the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is explored and processed. We will soon give more details about the method itself. The fact is that the method can be viewed as an entity evolving in the chamber structure and processes each chamber visited in order to produce some output. We believe that it is essential to approach things in a down-to-earth way and will therefore use a smiley to represent Borcherds' method as a hamster exploring and processing a chamber structure, like a hamster in a maze, except that our hamster obeys strict rules, described in section 1.7. The hamster in this illustration is pictured as located inside of the initial chamber, colored in green. We can therefore assume that Borcherds' method just started its execution. We start by focusing on how the method navigates within the chamber structure.

Internal procedures **DeltaW** and **SetOfWalls**, both introduced in section 1.5 enable Borcherds' method to compute the set of walls of a $\mathcal{P}_S$-chamber from the input data of its Weyl vector. When the set of walls of a chamber has been computed, Borcherds' method enforces the procedure **RatDetect** to identify walls associated with classes of self-intersection -2, which are usually referred to as (-2)-walls. Such walls, if crossed, would make Borcherds' method leave the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ and should therefore be avoided at all costs. When the data of the non (-2)-walls of a chamber D has been obtained, Borcherds' method is allowed to enter the chambers adjacent along these walls, that is, to cross the wall to enter the chamber adjacent along this wall to the chamber where it is currently located. The hamster located in the green chamber is thus allowed to visit the adjacent blue chambers as soon the non (-2)-walls of the green chamber are determined. Assume given a $\mathcal{P}_S$-chamber D such that the following data is available:

- The Weyl vector w_D of D
- A wall of the chamber D,

Using this data as input, the procedure **WeylAdj** introduced in section 1.7.2 outputs the Weyl vector w' of the chamber D' adjacent to D along the wall which has been specified in the input data. We have seen the basic principles governing Borcherds' method movement inside of the $\mathcal{P}_S$-chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$. A table listing all the procedures involved within Borcherds' method can be found by clicking here. Let us outline how the method processes the chambers it explores in order to fulfill its purpose, which consists in

Computing a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$.

Processing the chamber structure consists in using brute force flavored procedures in order to exhibit generators of

$$\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$$

from the data of the chambers explored by Borcherds' method. Generators are obtained in two ways:

- For each $\mathcal{P}_S$-chamber D explored, by computing a generating set of the group

$$\begin{aligned}\mathrm{Aut}_{\mathbf{H}}(D) = \{g \in \mathbf{H} \mid D^g = D\} \\ \subset \mathbf{H} \subset O^+(S)\end{aligned}$$

  of transformations in $\mathbf{H}$ preserving $D \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$.

To this end, the brute-force procedure **AutChamber** from section 1.7.3 takes as input the data of the walls of D and outputs a generating set of

$$\mathrm{Aut}_{\mathbf{H}}(D) \subset \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S).$$

- By testing whether two $\mathcal{P}_S$-chambers

$$D, D' \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$$

  are $\mathbf{H}$-congruent. That is, by determining whether there exists a transformation $g \in \mathbf{H}$ such that

$$D^g = D'.$$

Doing so is the purpose of the procedure **CongChecker** detailed in the section 1.7.4 of this document. This procedure takes as input the data of the respective sets of walls $\Omega(D)$ and $\Omega(D')$ of $\mathcal{P}_S$-chambers D, D' and determines whether the two chambers are $\mathbf{H}$-congruent. When this is the case, this brute-force flavored procedure returns at least one transformation establishing the $\mathbf{H}$-congruency.

That is, an element $g \in \mathbf{H}$ such that

$$\begin{aligned} D' &= D^g \\ &= \{xg \mid x \in D\} \end{aligned}$$

The **CongChecker** procedure has a central role within the algorithmic structure of the classical Borcherds' method. One of the innovations brought by this thesis is that, as we will see in section 1.11.1, the **CongChecker** congruence testing procedure is deployed in parallel over CongChecker blocks by using process-based parallelism, which yields huge performance improvements and led us to a modernization of Borcherds' method called the Poolized Borcherds' method, also introduced in section 1.11.1. Note that **CongChecker** and **AutChamber** both integrate a feature enabling them to test transformations for membership in $\mathbf{H}$. Knowledge of a membership criterion for $\mathbf{H}$ is therefore necessary. In his article [19], Shimada's approach to issues related to the membership criterion may lead his readers to think that it is necessary to handcraft a specific criterion for each surface on which Borcherds' method is to be applied, thus potentially discouraging people from venturing down this path. By studying the clues on this issue left by Shimada in [19], we provide in proposition 24 of section 1.6.2 a generalized membership criterion for $\mathbf{H}$. The result of this endeavor is the **MemberCrit** procedure, which takes as input the $(\rho \times \rho)$-sized matrix of a transformation generated by **CongChecker** or **AutChamber**, and determines whether it belongs to $\mathbf{H}$. Assume given a complex $K3$ surface X with Néron-Severi primitively embedded into a suitable even hyperbolic lattice and that Borcherds' method, which has not been discussed yet, has been executed and produced a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$.

What about the automorphism group of X?

We can now provide an answer to this fundamental interrogation. We start by denoting by T the transcendental lattice of X, that is, the orthogonal complement of $S = \mathrm{NS}(X)$ in

$$H^2(X, \mathbb{Z}) \simeq U^{\oplus 3} \oplus E_8(-1)^{\oplus 2}.$$

Consider the natural morphism

$$\eta_T : O(T) \longrightarrow O(T^\vee/T)$$

which realizes isometries of T as isometries of its discriminant group $T^\vee/T$. It turns out that if the complex $K3$ surface X under study satisfies

$$\rho_X < 20 \quad \text{and} \quad -1 \notin \mathrm{Ker}(\eta_T)$$

then there is an isomorphism

$$\mathrm{Aut}(X) \simeq \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S).$$

From the beginning, the logical structure leading to this result was contained in Shimada' article [19]. Obtaining this result amounted to assembling a jigsaw puzzle while always bearing in mind the goal of exhibiting a generalized framework of application for Borcherds' method. We were stunned that this result had not yet been explicitly formulated. However, more had still to be done. Such a result is worthless if one does not provide a general procedure to check whether

$$-1 \notin \mathrm{Ker}(\eta_T)$$

holds. Let us briefly explain how we proceeded in order to fill this gap. Before proceeding further, we have to mention that in case the above condition is not satisfied, i.e., when $-1 \in \mathrm{Ker}(\eta_T)$, then nothing prevents us from executing Borcherds' method. However, obtaining a generating set of $\mathrm{Aut}(X)$ is not guaranteed. For sure, we will obtain a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$, but

asserting anything about a generating set of $\mathrm{Aut}(X)$ when $-1 \in \mathrm{Ker}(\eta_T)$ is outside the scope of this thesis. We now go back to our initial discussion: Note that any embedding of S into

$$\mathbb{L} = U \oplus E_8(-1) \qquad \text{or into} \qquad \mathbb{L} = U \oplus E_8(-1) \oplus E_8(-1)$$

can be easily extended to an embedding of S into

$$H^2(X, \mathbb{Z}) \simeq U^{\oplus 3} + E_8(-1)^{\oplus 2}.$$

A Gram matrix G_T of the orthogonal complement of S into $H^2(X, \mathbb{Z})$ can then be easily obtained. Details and examples are provided on K3surfaces.com. In case the surface under study has Picard rank 18 or 19, obtaining an embedding of S into the K3 lattice from the data of the embedding of S into

$$\mathbb{L} = U \oplus E_8(-1) \oplus E_8(-1) \oplus E_8(-1)$$

is not guaranteed and this matter will have to be investigated on a case-by-case basis. Computing a Gram matrix G_T of the transcendental lattice T will thus be a straightforward job when the $K3$ surface under study has a Picard number less than or equal to 17 and has already been embedded into either

$$\mathbb{L} = U \oplus E_8(-1) \qquad \text{or} \qquad \mathbb{L} = U \oplus E_8(-1) \oplus E_8(-1).$$

Denote by $\mathrm{GL}_{22-\rho}(\mathbb{Z})$ the group of invertible $(22-\rho) \times (22-\rho)$-sized matrices with integer coefficients. The following criterion can be used to determine whether $-1 \notin \mathrm{Ker}(\eta_T)$ as soon as a Gram matrix G_T for the lattice T has been computed. We show in proposition 25 of section 1.6.3 that

$$2G_T^{-1} \notin \mathrm{GL}_{22-\rho}(\mathbb{Z}) \implies -\mathrm{Id} \notin \mathrm{Ker}(\eta_T)$$

and can therefore guarantee that the isomorphism

$$\mathrm{Aut}(X) \simeq \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$$

holds whenever the $K3$ surface under study has a transcendental lattice T with Gram matrix G_T satisfying

$$2G_T^{-1} \notin \mathrm{GL}_{22-\rho}(\mathbb{Z}).$$

Performing this check is the purpose of our procedure **KerChecker**, backed by proposition 25 from section 1.6.3. At the program level, everything is automated so that the user will never have to perform by hand the above-mentioned check for complex $K3$ surfaces with Picard number less than or equal to 17. We have made an overview of most of the procedures required to execute Borcherds' method. The following table provides a correspondence between Shimada' original procedures which have been outlined in his 2013 article and our modernized implementations of these procedures, which enabled us to produce a fully operational and automated version of Borcherds' method. We did not stop there and even raised the stakes, as we will discuss in section 1.11.

Ref. in this thesis	Ref. in Shimada's work
Procedure **DegenTest**, section 1.2	Criterion 5.9 in [19]
Procedure **EmbUpdater**, section 1.8	-
Procedure **RatDetect**, section 1.7.1	Algorithm 6.1 in [19]
Procedure **DeltaW**, section 1.5	Algorithm 5.8 in [19]
Procedure **SetOfWalls**, section 1.5	Algorithm 3.17 in [19]
Procedure **WeylAdj**, section 1.7.2	Algorithms 5.13 / 5.14 in [19]
Procedure **MemberCrit**, section 1.6	-
Procedure **AutChamber**, section 1.6	Algorithm 3.18 in [19]
Procedure **CongChecker**, section 1.6	Algorithm 3.19 in [19]
Procedure **ShiVectors**, section 1.4	Algorithm 2.1 in [18]
Procedure **KerChecker**, section 1.6.3	-

Note that a more detailed version of this table is available online.

A table describing all the procedures involved in Borcherds' method can be found by clicking here. All the procedures appearing in this table are fully detailed in this thesis, and we made sure to fill the gaps left in the wake of Shimada's 2013 article. We made sure to provide as much detail as possible. We now get to the heart of the matter and focus on Borcherds' method itself. Assume given an initial $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ with Weyl vector w_0 (the latter is fortunately provided by classical theory, see also the section 4, Vinberg-*Conway Theory*, from Shimada's article [19]) having the properties $\iota(S)$-nondegeneracy and inducing a $\mathcal{P}_S$-chamber contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. That is, the intersection

$$D_0 = \mathcal{D}_0 \cap \mathcal{P}_S$$

is a $\mathcal{P}_S$-chamber contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. In case all we have in hands is the data of a chamber $\mathcal{D}_0$ that does not satisfy the ι(S)-nondegeneracy property, we pick an ample class a_0 and make use of the procedure **EmbUpdater**, which has been mentioned earlier and is detailed in the section 1.8 of this thesis. If the program associated with the **EmbUpdater** procedure displays that another ample class should be chosen, it is recommended to do so and to execute **EmbUpdater** again. We thus assume that a transformation

$$\tau : \mathbb{L} \to \mathbb{L}$$

has finally been obtained and enables us to define an updated embedding

$$\tau \circ \iota : S \hookrightarrow \mathbb{L}$$

under which the $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ is $(\tau \circ \iota)\,(S)$-nondegenerate and satisfies

$$\mathcal{D}_0 \cap \mathcal{P}_S \subset \mathrm{Nef}(X) \cap \mathcal{P}_S.$$

Before proceeding further, note that in practice, to each $\mathcal{P}_S$-chamber is asso-

ciated a tuple that characterizes the chamber and provides data which can be processed within an implementation of Borcherds' method, as explained in section 1.7. Hence, a $\mathcal{P}_S$-chamber D is realized as a concrete data tuple such as

$$D = \big(w_D, \mathcal{A}_{\mathbf{H}}(D), \Omega(D), \overline{\Omega}(D)\big)$$

where w_D denotes the Weyl vector of D, where $\mathcal{A}_{\mathbf{H}}(D)$ denotes a generating set of $\mathrm{Aut}_{\mathbf{H}}(D)$, where $\Omega(D)$ denote the set of walls of D and where $\overline{\Omega}(D)$ denotes the set of walls of D with respect to anti-backtracking. More details about anti-backtracking can be found by clicking here. Note that our use of the term *classical* Borcherds' method refers to Shimada's original vision of Borcherds' method, for which he laid the algorithmic building blocks in his 2013 article [19], which has been a tremendous asset for us during our thesis.

We now explain the iterative mechanics behind the classical Borcherds' method. Full details are provided in section 1.7. Keeping this figure close by may be useful to the reader for what comes next. Note that the finiteness of the number of congruence classes of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is assured by Shimada, as indicated in his article [19], thus ensuring that Borcherds' method ends its execution at one moment or another. Fix a positive integer $k \neq 0$. We assume that Borcherds' method already performed k iterations and is currently at the beginning of its $(k+1)$-th iteration. For each positive integer j less than or equal to k, we thus assume that the method produced a set $\mathcal{L}_j$ containing chambers of level j, each representing their own $\mathbf{H}$-congruence class of chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$. For example,

$$\mathcal{L}_0 = \{D_0\}$$

since D_0 is by definition the only chamber of level 0, and is by default chosen as a representative of its $\mathbf{H}$-congruence class because it is the first chamber explored and processed by the method. Assume that the generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ which have been detected by the method during the previous iterations have been stored into a set Γ. The $(k+1)$-th iteration of Borcherds' method consists

in exploring and processing the chambers of level $k+1$ adjacent to the chambers in $\mathcal{L}_k$ along their non (-2)-walls in order to identify chambers representing new $\mathbf{H}$-congruence classes. Such chambers are stored into an initially empty set

$$\mathcal{L}_{k+1} = \{\,\}$$

and their adjacencies explored during the $(k+2)$-th iteration, provided that $\mathcal{L}_{k+1}$ is not empty at the end of the $(k+1)$-th iteration. Borcherds' method otherwise stops and returns all the data collected during its execution. For each chamber $D \in \mathcal{L}_k$, Borcherds' method detects the (-2)-walls among the elements of the set of walls of D by running the procedure **RatDetect** and classes in S associated with such walls are stored into the set $\mathcal{R}_{\text{rat}}$. As indicated earlier, we denote by $\overline{\Omega}(D)$ the set of walls of D taken with respect to anti-backtracking, i.e., the set $\Omega(D)$ from which the walls leading to chambers of level $k-1$ have been removed (click here for more details about anti-backtracking). For each $m \in \overline{\Omega}(D)$, the method uses the procedure **RatDetect** from section 1.7.1 to determine whether $(m)^{\perp}$ is a (-2)-wall. When $(m)^{\perp}$ is not a (-2)-wall, Borcherds' method computes the Weyl vector w' of the chamber D' adjacent to D along the wall $(m)^{\perp}$ by using the procedure **WeylAdj** from section 1.7.2 with the input of $m \in \Omega(D)$ and of the Weyl vector w of D. Note that $\Omega(D)$ can be taken modulo $\mathrm{Aut}_{\mathbf{H}}(D)$ before performing the computation of the Weyl vectors of adjacent chambers, thus saving resources in some cases. We have to mention that all our implementations of Borcherds' method possess this feature (quite easy to implement with GAP functions), but we deliberately omitted it from our structure diagrams so as not to burden them with a feature which, in practice, is not useful for cases where X has a small Picard number. Indeed, for such surfaces, which have been mainly studied during this thesis, the group $\mathrm{Aut}_{\mathbf{H}}(D)$ is almost systematically trivial for all chambers. This phenomenon has also been observed by Shimada ten years ago in [19]. Borcherds' method then computes the set of walls of D' by using the Weyl w' vector of D' as input into the procedure **DeltaW**, from section 1.5.1. It then uses the output of the latter into the procedure **SetOfWalls** from section 1.5, which returns the desired set $\Omega(D')$ of walls of the chamber D'.

Afterwards, the set of walls of D' is used as input into the procedure **AutChamber**, from section 1.7.3, which provides Borcherds' method with a set $\mathcal{A}_{\mathbf{H}}(D')$ of generators of

$$\mathrm{Aut}_{\mathbf{H}}(D') = \{g \in \mathbf{H} \mid D'^{g} = D'\}\,.$$

Note that such generators are also generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$, hence Borcherds' method stores them into the set Γ. Borcherds' method then determines whether D' represents a new $\mathbf{H}$-congruence class of chambers by proceeding as follows: For each chamber

$$D'' \in \mathcal{L}_0 \cup \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_k \cup \mathcal{L}_{k+1}$$

Borcherds' method uses the respective sets of walls of D' and of D'' as input data into the procedure **CongChecker** from section 1.7.4. The latter then uses brute-force to determine whether D' and D'' are $\mathbf{H}$-congruent. When the chambers D' and D'' are indeed $\mathbf{H}$-congruent, the procedure **CongChecker** provides at least one element $g \in \mathbf{H}$ establishing the congruence between D' and D''. Note that such transformations are generators of

$$\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$$

and are stored into the set Γ. If D' is not $\mathbf{H}$-congruent to a chamber in

$$\bigcup_{j=0}^{k+1} \mathcal{L}_j$$

then D' represents a new congruence class of chambers. Borcherds' method hence stores the data tuple

$$\big(w', \mathcal{A}_{\mathbf{H}}(D'), \Omega(D'), \overline{\Omega}(D)\big)$$

associated with the chamber D' into the set $\mathcal{L}_{k+1}$ which contains the chambers of level $k+1$ each representating a new congruence class discovering during the

current iteration, i.e. $(k+1)$-th iteration. When the chambers of level $k+1$ adjacent to chambers in $\mathcal{L}_k$ have all been explored and processed, two possibilities arise:

- If $\mathcal{L}_{k+1} \neq \emptyset$, that is, if representatives of new congruence classes have been detected during the iteration, then Borcherds' method proceeds to its next iteration: It explores and processes chambers of level $k+2$ adjacent to chambers in $\mathcal{L}_{k+1}$ by adjacency along non (-2)-walls.
- If $\mathcal{L}_{k+1} = \emptyset$, i.e., if no representative of new congruence classes have been detected during the iteration, then the methods ends and outputs all the data collected during its execution: Generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$, data of the (-2)-walls identified during the exploration, data of the representatives of congruence classes, which form a complete set of representatives of $\mathbf{H}$-congruence classes of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$.

Assume that the complex $K3$ surface under study satisfies $-1 \notin \mathrm{Ker}(\eta_T)$ and has Picard number $\rho_X < 20$, so that

$$\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S) \simeq \mathrm{Aut}(X)$$

holds, as indicated in theorem 22 from section 1.6. Assume moreover that the condition

$$\mathrm{Aut}_{\mathbf{H}}(D) = \{\mathrm{Id}\}$$

holds for all chambers D in the complete set of representatives returned by Borcherds' method. We show in proposition 31 from section 1.9 that the union of the set of chambers each representing their own congruence class returned by Borcherds' method is then a fundamental domain for the action of $\mathrm{Aut}(X)$ on $\mathrm{Nef}(X) \cap \mathcal{P}_S$. In proposition 37, we show that each orbit of smooth rational curves on X under the action of $\mathrm{Aut}(X)$ then possesses at least one representative among the classes in S associated with (-2)-walls contained in the set $\mathcal{R}_{\mathrm{rat}}$. The cardinality of $\mathcal{R}_{\mathrm{rat}}$ thus provides an upper bound on the number of orbits of smooth rational curves on X under the action of $\mathrm{Aut}(X)$. We provide

an algorithmic method in section 1.10 to identify redundant representatives in $\mathcal{R}_{\text{rat}}$, thus enabling us to refine this upper bound. We implemented a complete algorithmic suite for Borcherds' method in Python and made extensive use of mathematical functions from the SageMath library. In order to provide a framework of use that is accessible and familiar to most people, our programs can be launched from a simple Sage console. We did our best to put computer science at the service of pure mathematics. In this perspective, three fully functional instances of Borcherds' method arise from this thesis:

- The classical Borcherds' method is an implementation of the method that does not take advantage of the multi-core architecture of a CPU.

- The Poolized Borcherds' method is an upgrade of the classical Borcherds' method, which takes advantage of the multi-core architecture of the processor on which it is executed. Most of the procedures have been redesigned so that the workload through them can be distributed over several worker processes. To do so, we made use of Python's multiprocessing library. Note that running the Poolized Borcherds' method with the allocation of a single worker process amounts to running the *classical* Borcherds' method.

- We also implemented parallelism at the level of the method itself. What we did with the Poolized Borcherds' method consisted in adapting the internal procedures of the method so that process-based parallelism can then be used. However, enforcing parallelism at the level the method itself, e.g., by parallelizing the exploration of the chamber structure, requires more effort than revamping the code to deploy a solution such a **Pool**. We cover this in section 1.11, *Toward a parallelized Borcherds' method.*

Note that computers are tools, and that we always strive to make the best possible use of the tools at our disposal. However, the tools should not take over the content. This is why all the discussions in this thesis occur outside of the constrained framework of a particular language. Moreover, it should be noted

that not a single explicit reference to the code is used in this entire document. We believe that the classical dissertation format is not adapted to this aspect of our work. We instead provide an online platform on which we deal with all the practical and computer-based considerations: K3surfaces.com. Let us nevertheless conclude the introduction to the first part of our thesis on a very concrete and practical consideration: Let X be a complex $K3$ surface of Picard number inferior or equal to 17. Within this framework, we can guarantee full automation for all the procedures. The input data required to set up the environment which will enable us to execute the Borcherds method consists of:

- The data of a Gram matrix G_S of the Néron-Severi group $S := \mathrm{NS}(X)$ of the $K3$ surface X.
- The data of elements $v_1, \ldots, v_\rho \in \mathbb{L}$ such that the mapping defined by

$$\iota : [\alpha_1, \ldots, \alpha_\rho]_S \in S \longmapsto \alpha_1 v_1 + \cdots + \alpha_\rho v_\rho \in \mathbb{L}$$

  is a primitive embedding of S into either

$$\mathbb{L} = U \oplus E_8(-1) \qquad \text{or} \qquad \mathbb{L} = U \oplus E_8(-1) \oplus E_8(-1)$$

  depending of the Picard number of the $K3$ surface X under study.
- The data an ample class $a_0 := [\alpha_0, \ldots, \alpha_\rho]_S \in \mathrm{NS}(X)$.

The data of a list

$$[G_S, [v_1, \ldots, v_\rho], a_0],$$

where G_S is a $(\rho \times \rho)$-sized Sage matrix, where each v_i is a lattice vector of $\mathbb{L}$, and where a_0 is $(1 \times \rho)$-sized Sage matrix, is therefore all that is needed to execute our implementation of Borcherds' method.

```
ⓘ  Running the Poolized Borcherds' method with an allocation of 16 workers since :   0.05  minutes   /   0.0  hours

ⓘ  Slow mode enabled

--------------------------------------------------------------------------------------------------------------------------------

ⓘ  The Néron-Séveri Group of the surface under study has gram matrix :

[ 4  0  0]
[ 0 -2  0]
[ 0  0 -2]

ⓘ  The case under study is chafe_apparatus

ⓘ  There is 1 chamber of level 0 and a single congruence class of chambers has been discovered.

ⓘ  Currently discovering chambers of level 1 by adjacency to chambers of level 0 .

✔  There is no other chamber of level 0 to be processed during this level !

►  Generators of Aut(X) have not yet been detected...

--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------

ⓘ  Exploring the vicinity of the chamber D with Weyl vector [  31   30  -68  -46  -91 -135 -110  -84  -57  -29]  :

      ➡  Exploring the chamber with Weyl vector [ 31  30 -47 -46 -70 -95 -74 -54 -35 -17] adjacent to D along the wall [0 1 0]

         ↪  Testing whether this chamber is congruent to a chamber explored earlier...

                  ✔  Done in 0.0 seconds !

      ➡  Exploring the chamber with Weyl vector [  32   30  -56  -48  -80 -113  -91  -70  -50  -31] adjacent to D along the wall [ 1 -6 -2]

         ↪  Testing whether this chamber is congruent to a chamber explored earlier...

                  ✔  Done in 0.0 seconds !
```

```
►  The Néron-Séveri Group of the surface under study has Gram matrix :

[ 4  0  0]
[ 0 -2  0]
[ 0  0 -2]

►  A generating set of the automorphism group of the surface under study is given by :

[
[ 3 -4  0]  [ 3  0 -4]
[ 2 -3  0]  [ 0  1  0]
[ 0  0  1], [ 2  0 -3]
]

►  There are 10 congruence classes of chambers !

►  The Poolized Borcherds' method produced a fundamental domain of the action of Aut(X) onto Nef(X)∩ P(S) !

►  There are at most 2 orbits of smooth rational curves, with representatives : [[0 0 1], [0 1 0]]

--------------------------------------------------------------------------------------------------------------------------------

ⓘ  Poolized Borcherds' method executed with an allocation of 16 workers.

ⓘ  Total elapsed time (in hours) :    0.01 hours

ⓘ  Total elapsed time (in minutes) :    0.42 minutes.
```

More information and detailed examples are available online:

Guide: K3surfaces.com/aut-groups

Examples: K3surfaces.com/examples-borcherds

Let X be a complex $K3$ surface with Néron-Severi group $S = \mathrm{NS}(X)$. The other side of our study takes its roots in the fact that Roulleau produced a Magma program based on an algorithm due to Vinberg, which takes as input a Gram matrix of $\mathrm{NS}(X)$, an ample class P_0, integers $d, u_b \in \mathbb{Z}$, and outputs the set

$$\{C \in S \mid \langle C, C\rangle_S = d,\ \langle C, P_0\rangle_S \leq u_b\}$$

of classes of curves C of self-intersection $C^2 = d$ and for which the value of their intersection product with P_0 is less than or equal to u_b. We took advantage of Sage's interface to Magma in order to bring Roulleau's program directly into the practical world of Python. Combining this tool with Saint-Donat's & Morisson's results on projective models of $K3$ surfaces enabled us to study projective models of $K3$ surfaces with Néron-Severi group isomorphic to the integral lattice with Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

with respect to a fixed basis, and even discuss the unirationality of their moduli spaces. Here again, the tools produced during our thesis to do so have a scope of application which extends far beyond these $K3$ surfaces. Our **PModChecker** program bears witness to this fact. The fact is that the computer-based algorithmic approach we adopted led us to produce innovative tools. For example, we combined various algorithmic pieces provided by Shimada in his article [18] in order to produce:

- **Universal ampleness tester for classes of divisors on $K3$ surfaces**: Given an ample class $a_0 \in S$ and a Gram Matrix G_S for S, our program **AmpTester** can determine whether any class $D \in S$ is ample or not. This program makes use of algorithmic material due to Shimada [18].

```
sage: AmpTester(Matrix([50,20,40]))

    ⓘ The known ample class amp0 is [ 2 -1 -1]

    ⓘ The K3 surface under study has Néron-Severi lattice with Gram matrix :

[ 4  0  0]
[ 0 -2  0]
[ 0  0 -2]

********************************************************************************

    ⚠ The class [50 20 40] is NOT AMPLE !

    ⚠ Note that the (-2)-curves contained in

[[0 0 1], [0 1 0]]

       are obstructions to the ampleness of [50 20 40] !

    ⚠ Indeed, their intersection product with the class [50 20 40] is strictly negative !

    ⚠ AmpTester Boolean value is : False
```

The following theorem incorporates results from Saint-Donat [17] & Morrison and can be found in the latter's 1988 Cortona lectures [13] and provides characterizations of the projective models which can be obtained from the data of an ample class on a $K3$ surface. We state it in its formulation by Debarre in his Master's course [3, Section 3.4]:

SDM Theorem. Let X be a $K3$ surface and $D \in S$ an ample class.

(a) If $D^2 = 2$ and there does not exist a class $F \in \mathrm{NS}(X)$ such that $F^2 = 0$ and $F \cdot D = 1$ then $\varphi_D : X \longrightarrow \mathbb{P}^2$ is a double cover.

(b) If $D^2 = 4$ and there does not exist a class $F \in \mathrm{NS}(X)$ on X such that $F^2 = 0$ and $F \cdot D \in \{1, 2\}$ then $\varphi_D : X \longrightarrow \mathbb{P}^3$ embeds X as a quartic surface in $\mathbb{P}^3$.

(c) If $D^2 = 6$ and there does not exist a class F on X such that $F^2 = 0$ and $F \cdot D \in \{1, 2\}$ then $\varphi_D : X \longrightarrow \mathbb{P}^4$ embeds X as a degree 6 surface in $\mathbb{P}^4$.

(d) If $D^2 = 8$ and there does not exist a class F on X such that $F^2 = 0$ and $F \cdot D \in \{1, 2, 3\}$ then $\varphi_D : X \longrightarrow \mathbb{P}^5$ either embeds X as a generically transverse intersection of three quadrics in $\mathbb{P}^5$ with only rational double points, or φ_D realizes X as double cover of a Veronese surface.

This theorem led us to produce and implement the following tool:

- **PModChecker** (**SDM** theorem tester): Given an ample class $a_0 \in S$ and a Gram Matrix G_S for S, our program **PModChecker** can determine whether a given class $D \in S$ can enter within the framework of the above-mentioned **SDM** theorem. When this is the case, **PModChecker** specifies which projective model of the $K3$ surface under study can be obtained thanks to the map into projective space associated with φ_D, in virtue of the Saint-Donat / Morrison Theorem. This program extensively relies on an algorithmic routine which was originally intended for other purposes and can be found in Shimada's article [18].

```
sage: GramMatS
[14  0  0]
[ 0 -2  0]
[ 0  0 -2]
sage: PModChecker(Matrix([1,-2,-1]))

      ⓘ The class under study is  D = [ 1 -2 -1]

      ✔ The class  D = [ 1 -2 -1]  is ample and has self-intersection 4 !

      ✔ There does not exist classes F ∈ NS(X) such that F.D ∈{1,2} and F.F = 0 !

      ↪ By the Saint-Donat / Morrison Theorem, the map  φ_D : X  ⟶  P^3  embeds X as a quartic in P^3 !

sage: PModChecker(Matrix([450,50,-20]))

      ⓘ The class under study is  D = [450  50 -20]

      ⚠ The class  [450  50 -20]  is NOT ample !

      ⚠ This class cannot be associated with a projective model of the K3 under study !

sage:
```

More information and detailed examples are available online:

AmpTester: K3surfaces.com/amptester

PModChecker: K3surfaces.com/pmodchecker

Table of Contents

Contents

Part I
Automorphisms groups and orbits of smooth rational curves on $K3$ surfaces

1 Automorphism groups and orbits of (-2)-curves

1.1 Generalities

The following section introduces the main theoretical tools, notions, and concepts with which the reader should be familiar before pursuing the study further.

1.1.1 The basics

We recall that a free $\mathbb{Z}$-module L of finite rank with a non-degenerate symmetric bilinear form

$$\langle\ ,\ \rangle_L : L \times L \longrightarrow \mathbb{Z}$$

is called an *integral lattice*. In the following, we will use the term *lattice* to refer to an integral lattice. A lattice L is said to be *even* if

$$x^2 := \langle x, x \rangle_L \in 2\mathbb{Z}$$

holds for any lattice element $x \in L$. The Gram matrix of a lattice L of rank N with basis $b_1, \ldots, b_N$ is defined as the matrix

$$G_L = \left[\langle b_i, b_j \rangle_L\right]_{1 \leq i,j \leq N}$$

Denote by n_+ the number of positive eigenvalues and by n_- the number of negative eigenvalues of G_L. The pair of integers (n_+, n_-) is called the *signature* of L. A lattice L is said to be *hyperbolic* when $L \otimes \mathbb{R}$ is of signature $(1, n-1)$. The *determinant* of a lattice L is defined as the determinant of the Gram matrix G_L of L . An *unimodular* lattice is an integral lattice of determinant ± 1. Let L be an hyperbolic lattice. One of the two connected components of the set

$$\left\{x \in L \otimes \mathbb{R} \mid x^2 > 0\right\}$$

is called a *positive cone* of L and is denoted by $\mathcal{P}_L$. It inherits the topology from the vector space $\mathbb{L} \otimes \mathbb{R}$. When the lattice under study is chosen to be the Néron-

Severi lattice $S := \mathrm{NS}(X)$ of a $K3$ surface X, the positive cone $\mathcal{P}_S$ is chosen as the connected component of

$$\left\{x \in S \otimes \mathbb{R} \mid x^2 > 0\right\}$$

containing ample classes. Denote by $O(L)$ the group of isometries of a lattice L. We view elements of $O(L)$ as matrix transformations of size $\mathrm{rank}(L) \times \mathrm{rank}(L)$ and use the convention that elements of L are represented as row vectors of size $\mathrm{rank}(L)$. That is, the image of an element $v \in L$ by a transformation $g \in O(L)$ is a row vector of size $\mathrm{rank}(L)$ and given by

$$v \longmapsto vg.$$

Representing elements of L as row vectors instead of column vectors may seem a bit unusual in a classical setting. It is, however, perfectly suitable when working with a CAS such as Magma or Sage, in which lattice elements are realized as row vectors. Regarding elements of $O(L)$, note that an invertible matrix g is the matrix of a transformation of $O(L)$ if and only if it preserves the bilinear form, that is, if and only if

$$gG_Lg^T = G_L$$

holds. The stabilizer subgroup of the positive cone $\mathcal{P}_L$ in $O(L)$ is denoted by $O^+(L)$. Let L be an even lattice. An element $r \in L$ such that $r^2 = -2$ is called a *root*. To each root $r \in L$ can be associated a *reflection*

$$s : L \longmapsto L$$

defined by

$$s_r : x \longmapsto x + \langle x, r\rangle r.$$

Note that s_r is an involution. That is,

$$s_r \circ s_r = \mathrm{Id}$$

holds. The subgroup of $O^+(L)$ generated by all the reflections s_r with respect to the roots is denoted by $W(L)$ and called the *Weyl Group* of L. The quotient $L^\vee/L$ is called the discriminant group of the lattice L. The discriminant group is endowed with a non-degenerate quadratic form

$$q : L^\vee/L \longrightarrow \mathbb{Q}/2\mathbb{Z}$$

defined by

$$q : x \bmod L \longmapsto x^2 \bmod 2\mathbb{Z}.$$

The form q_L is called the discriminant form of L. We use the notation $(L^\vee/L, q_L)$ in order to refer to the discriminant group and to its associated quadratic form at the same time. The group of isometries of $(L^\vee/L, q)$ is denoted by $O(q_L)$. There is a natural homomorphism

$$\eta : O(L) \to O(q_L)$$

between the group of isometries of L and the group of isometries of $(L^\vee/L, q_L)$.

1.1.2 Chamber structure and walls

Let L be an even hyperbolic lattice and let $\mathcal{P}_L$ be a positive cone of L

Definition 1. Let $\Delta \subset L$. The set

$$\Sigma_L(\Delta) = \{x \in L \otimes \mathbb{R} \mid \forall v \in \Delta,\, \langle x, v\rangle_L \geq 0\},$$

is called the positive cone associated with Δ.

It is also referred to as the Δ-positive cone. Define

$$\mathcal{N}_L = \{x \in L \otimes \mathbb{R} \mid \langle x, x\rangle_L < 0\}.$$

A closed subset D of $\mathcal{P}_L$ is called a *chamber* if it has non-empty interior and if

there exists a subset $\Delta \subset \mathcal{N}_L$ such that

$$D = \Sigma_L(\Delta) \cap \mathcal{P}_L. \tag{1.1}$$

Such a subset Δ is called a *defining set* of the chamber D. Note that the definition of a chamber does not prohibit the fact that a chamber can be associated with more than one defining set. Keeping this fact in mind is necessary to understand the path leading to the notion of *set of walls* of a chamber, introduced in section 1.1.2 of this thesis.

Definition 2. A subset $\Delta \subset \mathcal{N}_L$ is a called a *defining set* of a chamber D whenever the equality $D = \Sigma_L(\Delta) \cap \mathcal{P}_L$ holds.

That is, an element $x \in \mathcal{P}_L$ is contained in a chamber $D = \Sigma_L(\Delta) \cap \mathcal{P}_L$ if and only if the inequalities

$$\langle x, v \rangle_L \geq 0$$

for all $v \in \Delta$.

Definition 3. Let $v \in L \otimes \mathbb{R}$. We denote by $(v)^\perp$ the orthogonal complement in $(L \otimes \mathbb{R}) \cap \mathcal{P}_L$ of the element v . That is,

$$(v)^\perp := \{x \in L \otimes \mathbb{R} \mid \langle x, v \rangle_L = 0\} \cap \mathcal{P}_L.$$

We recall that a collection of subsets of a topological space is said to be *locally finite* if each point of the space has a neighborhood intersecting only finitely many sets in the collection. Let $\mathcal{F} \subset \mathcal{N}_L$ be a subset such that the collection

$$\left\{(v)^\perp \mid v \in \mathcal{F}\right\}$$

of orthogonal complements in $\mathcal{P}_L$ of elements of $\mathcal{F}$ is a locally finite collection in $\mathcal{P}_L$. The positive cone $\mathcal{P}_L$ of the lattice L can be decomposed as follows:

$$\mathcal{P}_L = (\mathcal{P}_L \setminus \bigcup_{v \in \mathcal{F}} (v)^\perp) \cup \bigcup_{v \in \mathcal{F}} (v)^\perp. \tag{1.2}$$

Let C be a connected component of

$$(\mathcal{P}_L \setminus \bigcup_{v\in\mathcal{F}} (v)^\perp) = \bigcap_{v\in\mathcal{F}} \mathcal{P}_L \setminus (v)^\perp.$$

Then there exists a subset $\Delta_C \subset \mathcal{F}$ such that an element $p \in \mathcal{P}_{\mathbb{L}}$ belongs to C if and only if the strict inqualities

$$\langle p, v\rangle_L \;\; > \;\; 0$$

are satisfied for all $v \in \Delta_C$. Similary, if we denote by $\overline{C}$ the topological closure of C then an element $p \in \mathcal{P}_L$ belongs to $\overline{C}$ if and only if

$$\langle p, v\rangle_L \geq 0$$

holds for all $v \in \Delta_C$. We hence see that $\overline{C}$ can be expressed as

$$\overline{C} = \Sigma_L(\Delta_C) \cap \mathcal{P}_L,$$

where

$$\Sigma_L(\Delta_C) = \{x \in L \otimes \mathbb{R} \mid \forall v \in \Delta_C,\ \langle v, x\rangle_L \geq 0\}.$$

In virtue of definition 1, the set $D := \overline{C}$ is a chamber. We thus obtained:

Proposition 4. *The closure $\overline{C}$ in $\mathcal{P}_L$ of a connected component C of*

$$\mathcal{P}_L \setminus \bigcup_{v\in\mathcal{F}} (v)^\perp$$

is a chamber. Moreover, there exists a finite subset $\Delta \subset \mathcal{F}$ such that $\overline{C} = \Sigma_L(\Delta) \cap \mathcal{P}_L$. To any chamber $D = \Sigma_L(\Delta) \cap \mathcal{P}_L$ with $\Delta \subset \mathcal{F}$ can be associated a connected component C of

$$\mathcal{P}_L \setminus \bigcup_{v\in\mathcal{F}} (v)^\perp$$

such that $D = \overline{C}$.

We now assume fixed a subset $\mathcal{F} \subset \mathcal{N}_L$ having the property that

$$\left\{(v)^\perp \mid v \in \mathcal{F}\right\}$$

is locally finite.

Definition 5. The collection

$$\mathcal{C}_{\mathcal{F}} = \left\{ \overline{C} := \Sigma_L(\Delta_C) \cap \mathcal{P}_L \mid C \text{ connected component of } \mathcal{P}_L \setminus \bigcup_{v \in \mathcal{F}} (v)^\perp \right\}$$

is called a chamber structure on the positive cone $\mathcal{P}_L$ of the lattice L.

Assume that a chamber structure $\mathcal{C}_{\mathcal{F}}$ has been set on $\mathcal{P}_L$. We now introduce the important notion of *walls* of a chamber. Denote by $\mathrm{Int}(D)$ the topological interior of a chamber D of $\mathcal{C}_{\mathcal{F}}$.

Definition 6. a hyperplane $(v)^\perp$ of $\mathcal{P}_L$, with $v \in \mathcal{F}$, is called a *wall* of the chamber D whenever both of the following conditions are satisfied: *(a)* the equality

$$\mathrm{Int}(D) \cap (v)^\perp = \emptyset$$

holds and *(b)* there exists a non-empty open subset of $(v)^\perp$ contained in $D \cap (v)^\perp$.

For any chamber D, the inclusion

$$\left\{(v)^\perp \mid v \text{ is a wall of } D\right\} \subseteq \left\{(v)^\perp \mid v \in \Delta\right\}$$

always holds for any defining set Δ of D, will often happen to be a proper inclusion.

Two facts should be underlined:

- A defining set of a chamber D may contain elements that do not have an orthogonal complement defining a wall of the chamber D.

- Distinct elements of a defining set of a chamber may have the same orthogonal complement, and could thus define the same wall of D.

Let us have a short discussion about the fact that a defining set of a chamber can also contain elements having the same orthogonal complement in $\mathcal{P}_L$. In practice, the fact that $x_1 = \eta x_2$ for some $\eta \in \mathbb{Z}$ will often turn out to be the cause of such a situation. Let us show how to deal with elements related by such a relation. Assume that $\mathrm{rank}(L) = N$ for some integer $N > 0$ and fix a basis for L. We express elements of L in terms of their coordinates with respect the chosen basis. We let

$$x_1 = [\alpha_1, \alpha_2, \ldots, \alpha_N]_L \qquad \text{and} \qquad x_2 = [\beta_1, \beta_2, \ldots, \beta_N]_L$$

be distinct elements of L belonging to the defining set of some chamber, where the α_i and β_i for $1 \leq i \leq N$ are the respective coordinates of x_1 and x_2 with respect to the chosen basis of L. If we assume that $x_1 = \eta x_2$ for some integer $\eta \in \mathbb{Z}$, then the equality

$$\frac{x_1}{\gcd(\alpha_1, \alpha_2, \ldots, \alpha_N)} = \frac{x_2}{\gcd(\beta_1, \beta_2, \ldots, \beta_N)}.$$

obviously holds. Thus, the issue caused by the presence of elements such as $x_1$and x_2 in Δ can be overcome by replacing the latter by

$$\Delta' = \left\{ \frac{x}{\gcd(\gamma_1, \ldots, \gamma_N)} \mid x := [\gamma_1, \ldots, \gamma_N]_L \in \Delta \right\}. \tag{1.3}$$

which is obtained by dividing each element $x \in \Delta$ by the greatest common divisor of its coordinates. Given a chamber D, it would be very convenient if we could associate a set $\Omega(D)$ containing the elements of L which induce walls of

the chamber D. From what we just discussed, the set $\Omega(D)$ should by definition possess the two following properties:

- If $x \in \Omega(D)$, then $(x)^{\perp}$ is a wall of D.
- No two distinct elements $x_1, x_2 \in \Omega(D)$ should have the same orthogonal complement.

In the framework of the classical theory presented by Shimada in [19], defining sets possessing these two properties are called *minimal defining sets.*

Definition 7. A defining set Δ of a chamber D is said to be a *minimal defining set* whenever the two following conditions are satisfied:

(i) For all $x \in \Delta,$ the orthogonal complement $(x)^{\perp}$ is a wall of D.

(ii) Whenever $x, y \in \Delta$ are distinct, then $(x)^{\perp} \neq (y)^{\perp}$.

The next question that comes naturally is minimality. The terminology from the classical theory is quite misleading, because the definition of a minimal defining set does not insure true minimality. Indeed, note that in case no genuine minimality condition is incorporated into the definition of a set of walls, then any minimal defining set of a chamber could be taken as the *set of walls* of a chamber. For example, assume that $\{a, b, c, d\}$ is a minimal defining set of a chamber D. Then the set $\{99a, b, 40c, 28d\}$ is also a minimal defining set of D. As we discussed earlier in this section, setting up a chamber structure requires a set $\mathcal{F} \subset \mathcal{N}_L$ having the property that the associated collection of hyperplanes

$$\left\{(v)^{\perp} \mid v \in \mathcal{F}\right\}$$

is locally finite. We have seen that chamber structure $\mathcal{C}_{\mathcal{F}}$ is then obtained by taking the closure of each connected component of

$$\mathcal{P}_L \setminus \bigcup_{v \in \mathcal{F}} (v)^{\perp}.$$

The walls of the elements of the chamber structure thus originate from respective orthogonal complements in $\mathcal{P}_{\mathbb{L}}$ of elements of $\mathcal{F}$. It would thus be convenient to require that the elements in the set of walls $\Omega(D)$ of a chamber $D \in \mathcal{C}_{\mathcal{F}}$ are elements of $\mathcal{F}$. Fulfilling this requirement is the reason why the classical theory, found in Shimada's article [19], introduces the notion of $\mathcal{F}$-minimal defining set to take this fact into account, as described in [19].

Definition 8. A minimal defining set $\Delta \subset \mathcal{N}_L \subset L \otimes \mathbb{R}$ of a chamber D satisfying the conditions

(i) $\Delta \subset \mathcal{F}$,

(ii) if $x \in \Delta$, then $\alpha x \notin \mathcal{F}$ for all $0 < \alpha < 1$.

is called a *$\mathcal{F}$-minimal defining set* of D and is denoted by $\Delta_{\mathcal{F}}(D)$.

Assume that Δ is a defining set of a chamber D and that $\Delta \subset \mathcal{F}$.

In order to turn Δ into a minimal defining set, we apply definition 7. First, we have to make sure that no two distinct elements of Δ have the same orthogonal complement. The first step that should be taken in order to reach this goal consists in taking the set Δ' instead of Δ, where the former has been defined in expression 1.3. It should be noted that in spite of our assumption $\Delta \subset \mathcal{F}$, there is absolutely no guarantee that $\Delta' \subset \mathcal{F}$ will also hold. The best way to deal with this issue consists in requiring that the set of walls $\Omega(D) \subset L \otimes \mathbb{R}$ of a chamber D has the property that its elements cannot be expressed as integer multiples of other elements of $\mathbb{L} \otimes \mathbb{R}$. In order to do so, it is convenient to use the fact that an integral lattice such as L is naturally contained in its dual lattice $L^{\vee}$, thus enabling us to work directly within the framework of dual lattices in which the requirement mentioned above can always be fulfilled.

The classical theory built, by Shimada in [19], embodies all these considerations by introducing of the notion of *primitively minimal defining set*.

Definition 9. A minimal defining set Δ of an $\mathcal{F}$-chamber D such that every $v \in \Delta$ is primitive in $L^\vee$ is called a *primitively minimal defining set* of D.

In this thesis, the term **set of walls** refers to a primitively minimal defining set. That is, a sentence such as

$$\Delta \textit{ is a primitively minimal defining set of } D$$

from Shimada's classical theory thus becomes

$$\Delta \textit{ is the set of walls of } D$$

in the framework of our thesis. The notion of a set of walls will come up repeatedly throughout this thesis, and will be central during our entire study. Please remember that the notation $\Omega(D)$ denotes the set of walls of a chamber D.

It should be noted that our use of the term *set of walls* is an abuse of language. Indeed, the set of walls of a chamber, defined according to its name, should be defined as

$$\left\{(v)^\perp \mid v \text{ is a wall of } D\right\}$$

with additional minimality conditions, as discussed above. Our justification for this abuse lies in the fact that we adopt a computer-based algorithmic approach: Entities involved in the procedures must therefore be defined so that a computer can process them. Given a minimal defining set of a chamber D, we, therefore, explain in section 1.5.2 the mechanics behind our version of a procedure originating from Shimada's article [19] to compute the set $\Omega(D)$ of walls of a chamber D in a practical way. We close this section by asking our readers to keep in mind that, in practice, the first step leading to the set of walls $\Omega(D)$ from a defining set Δ of D consists in computing Δ' (see expr. (1.3)).

1.2 Induced chamber structure

In this section, we show that a chamber structure on the positive cone $\mathcal{P}_S$ of a $K3$ surface X is obtained whenever the two following conditions are satisfied:

- The Néron-Severi group S of X has been primitively embedded into an even hyperbolic lattice $\mathbb{L}$ in such a way that $\mathcal{P}_S \subset \mathcal{P}_S$.

- A chamber structure has been set on $\mathcal{P}_\mathbb{L}$ by taking $\mathcal{F} = \mathcal{R}_\mathbb{L}$ in the definition 5 of a chamber structure.

Let X be a complex $K3$ surface. As before, we denote by $S = \mathrm{NS}(X)$ its Néron-Severi group and let $\rho_X = \mathrm{rank}(S)$ denote the Picard number of X. We assume that S is primitively embedded into a suitable even hyperbolic lattice $\mathbb{L}$ chosen according to the value of ρ_X, as indicated in the following table:

Picard number ρ_X	Recommended ambient lattice
$1 \leq \rho_X < 10$	$U \oplus E_8(-1)$
$10 \leq \rho_X < 18$	$U \oplus E_8(-1) \oplus E_8(-1)$
$18 \leq \rho_X < 20$	$U \oplus E_8(-1) \oplus E_8(-1) \oplus E_8(-1)$

We moreover assume that the embedding

$$\iota : S \hookrightarrow \mathbb{L}$$

is such that the inclusion $\mathcal{P}_S \subset \mathcal{P}_\mathbb{L}$ holds. As before, we denote by

$$\mathcal{R}_\mathbb{L} = \{x \in \mathbb{L} \mid \langle x, x \rangle_\mathbb{L} = -2\}$$

the set of (-2)-vectors of $\mathbb{L}$. The local finiteness of the collection

$$\left\{(x)^{\perp} \mid \langle x, x\rangle_{\mathbb{L}} = -2\right\}$$

is established in Shimada's article [19, Lemma 3.4] . We thus apply definition 1.1.2 with $\mathcal{F} = \mathcal{R}_{\mathbb{L}}$ in order to obtain a chamber structure

$$\mathcal{C}_{\mathcal{R}_{\mathbb{L}}} = \{\bar{A} \subset \mathcal{P}_{\mathbb{L}} \mid A \text{ is a connected component of } \mathcal{P}_{\mathbb{L}} \setminus \bigcup_{v \in \mathcal{R}_{\mathbb{L}}} (v)^{\perp}\}$$

on the positive cone $\mathcal{P}_{\mathbb{L}}$ of the lattice $\mathbb{L}$. Chambers of this chamber structure will be referred to as $\mathcal{P}_{\mathbb{L}}$-chambers. In order to identify $\mathcal{P}_{\mathbb{L}}$-chambers, we will always make use of the **mathcal** font with the capital letter $\mathcal{D}$ and a numeral as a subscript when necessary. As indicated in the short introduction to this section, we will soon explain how a chamber structure on $\mathcal{P}_{\mathbb{L}}$ can induce a chamber structure on $\mathcal{P}_S$. Chambers belonging to the induced chamber structure on $\mathcal{P}_S$ will be referred to as $\mathcal{P}_S$-chambers, and such chambers will be denoted by using the standard font with a capital D. Denote by $R = S^{\perp}$ the orthogonal complement of S into $\mathbb{L}$. More generally, we use many notational conventions exactly as Shimada introduced them in his 2013 article Consider the orthogonal projections

$$\mathrm{pr}_S : \mathbb{L} \otimes \mathbb{R} \longrightarrow S \otimes \mathbb{R} \qquad \text{and} \qquad \mathrm{pr}_R : \mathbb{L} \otimes \mathbb{R} \longrightarrow R \otimes \mathbb{R}$$

from $\mathbb{L} \otimes \mathbb{R}$ to $S \otimes \mathbb{R}$ and from $\mathbb{L} \otimes \mathbb{R}$ to $R \otimes \mathbb{R}$, respectively. When appropriate, we will make use of the shorthand notations x_S and x_R to denote images of an element $x \in L \otimes R$ via the maps pr_S and pr_R defined above.

Proposition 10. *An element $x \in \mathcal{R}_{\mathbb{L}}$ such that $x_S \neq 0$ satisfies $(x)^{\perp} \cap \mathcal{P}_S \neq \emptyset$ if and only if*

$$\langle x_S, x_S\rangle_S < 0.$$

Proof. Before proceeding, we recall given $x \in \mathbb{L}$, we define

$$(x)^{\perp} = \{y \in \mathbb{L} \mid \langle x, y \rangle_{\mathbb{L}} = 0\} \cap \mathcal{P}_{\mathbb{L}}.$$

Let $y \in (x)^{\perp} \cap \mathcal{P}_S$. Since

$$y \in \mathcal{P}_S \subset S$$

we have

$$y = y_S \qquad \text{and} \qquad \langle y_S, y_S \rangle_S > 0.$$

We obtain

$$\langle x, y_S \rangle_{\mathbb{L}} = 0$$

using the fact that $y_S \in (x)^{\perp}$. Expressing the element x as $x_R + x_S$ then yields the equality

$$\langle x_S + x_R, y_S \rangle_S = 0.$$

from which we immediately obtain

$$\langle x_S, y_S \rangle_S = 0$$

where x_R is the projection of the element x onto $R = S^{\perp}$. By the Hodge Index theorem, this equality implies that

$$\langle x_S, x_S \rangle_S < 0.$$

To establish the converse, we now assume that this inequality holds. The orthogonal complement in S of x_S is then an hyperbolic lattice: It has rank $\rho - 1$ and signature $(1, \rho - 2)$. Thus, there exists an element in the orthogonal complement of x_S with strictly positive self-intersection. Such an element then clearly belongs to

$$(x_S)^{\perp} \cap \mathcal{P}_S,$$

and enables us to assert the non-emptiness of this set. Let $\mathcal{D}$ be a $\mathcal{P}_{\mathbb{L}}$-chamber with $\mathcal{R}_{\mathbb{L}}$-minimal defining set $\Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}) \subset \mathcal{R}_{\mathbb{L}}$. By definition 1 of a chamber, the

equality

$$\mathcal{D} = \Sigma_{\mathbb{L}}(\Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D})) \cap \mathcal{P}_{\mathbb{L}} \tag{1.4}$$

holds, where we recall that

$$\Sigma_{\mathbb{L}}(\Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D})) = \{y \in \mathbb{L} \mid \forall r \in \Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}),\ \langle y, r\rangle_{\mathbb{L}} \geq 0\}\,. \tag{1.5}$$

We now introduce the fundamental concept of *Weyl vector* of a $\mathcal{P}_{\mathbb{L}}$-chamber which originates from [19]. □

Definition 11. Let $\mathcal{D}$ be a $\mathcal{P}_{\mathbb{L}}$-chamber. An element $w \in \mathbb{L}$ is said to be a *Weyl vector of* $\mathcal{D}$ *if its* $\mathcal{R}_{\mathbb{L}}$*-minimal defining set* $\Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D})$ *is given by*

$$\Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}) = \{r \in \mathcal{R}_{\mathbb{L}} \mid \langle w, r\rangle_{\mathbb{L}} = 1\}\,.$$

Note that the definition 1 of a chamber implies that no two distinct chambers can have the same defining set. Since a minimal defining set is a defining set, it is clear that no two distinct chambers can have the same minimal defining set. A Weyl vector thus uniquely characterizes a single chamber. We will see in the upcoming sections that the knowledge of the Weyl vector of a chamber enables us to obtain precious information about the chamber such as its set of walls. Let $\mathcal{D}$ be a $\mathcal{P}_{\mathbb{L}}$-chamber with Weyl vector $w \in \mathbb{L}$ and assume that $\mathcal{D} \cap \mathcal{P}_S$ has a non-empty interior. We now show that this intersection can expressed as

$$\mathcal{D} \cap \mathcal{P}_S = \Sigma_S(\mathrm{pr}_S(\Delta_w)) \cap \mathcal{P}_S$$

for some set Δ_w depending on the Weyl vector of $\mathcal{D}$. Note that the right-hand side of this equality defines a chamber of $\mathcal{P}_S$ whenever it has a non-empty interior. This result will pave the way toward a definition of $\mathcal{P}_S$-chambers as chambers of $\mathcal{P}_S$ obtained by intersecting $\mathcal{P}_{\mathbb{L}}$-chambers with $\mathcal{P}_S$ provided that the resulting intersections have a non-empty interior. We then see that this definition enables us to obtain a $\mathcal{P}_S$-chamber structure from a $\mathcal{P}_{\mathbb{L}}$-chamber structure.

Using expressions (1.4) and (1.5), we see that $\mathcal{D} \cap \mathcal{P}_S$ can be expressed as

$$\mathcal{D} \cap \mathcal{P}_S = \{y \in \mathbb{L} \otimes \mathbb{R} \mid \forall r \in \Delta_{\mathcal{R}_\mathbb{L}}(\mathcal{D}),\ \langle y, r \rangle_\mathbb{L} \geq 0\} \cap \mathcal{P}_S.$$

The assumption that it has non-empty interior enables us to express the above equality as

$$\mathcal{D} \cap \mathcal{P}_S = \{y \in \mathbb{L} \otimes \mathbb{R} \mid \forall r \in \Delta_w,\ \langle y, r \rangle_\mathbb{L} \geq 0\} \cap \mathcal{P}_S$$

where Δ_w is defined as

$$\Delta_w = \left\{x \in \Delta_{\mathcal{R}_\mathbb{L}}(\mathcal{D}) \mid (x)^\perp \cap \mathcal{P}_S \neq \emptyset\right\}.$$

Note that this set is non-empty whenever $\mathcal{D} \cap \mathcal{P}_S$ has non-empty interior and that the equality

$$\langle y, x \rangle_\mathbb{L} = \langle y, \mathrm{pr}_S(x) \rangle_S$$

holds for all $y \in S$ and all $x \in \mathbb{L}$. Thus,

$$\mathcal{D} \cap \mathcal{P}_S = \{y \in \mathbb{L} \otimes \mathbb{R} \mid \forall r \in \Delta_w,\ \langle y, r_S \rangle_S \geq 0\} \cap \mathcal{P}_S.$$

We then have

$$\begin{aligned} \mathcal{D} \cap \mathcal{P}_S &= \{y \in \mathbb{L} \otimes \mathbb{R} \mid \forall r \in \Delta_w,\ \langle y, r \rangle_\mathbb{L} \geq 0\} \cap \mathcal{P}_S \\ &= \{y \in \mathbb{L} \otimes \mathbb{R} \mid \forall r \in \mathrm{pr}_S(\Delta_w),\ \langle y, r_S \rangle_S \geq 0\} \cap \mathcal{P}_S \end{aligned} \tag{1.6}$$

where we recall that x_S is a shorthand for the orthogonal projection $\mathrm{pr}_S(x)$ of an element of $\mathbb{L}$ onto S. We then note that the set $\Sigma_S(\mathrm{pr}_S(\Delta_w))$ is by definition defined as

$$\Sigma_S(\mathrm{pr}_S(\Delta_w)) = \{y \in S \otimes \mathbb{R} \mid \forall r \in \mathrm{pr}_S(\Delta_w),\ \langle y, r_S \rangle_S \geq 0\}.$$

Also, note that expression (1.6) is obviously equivalent to

$$\mathcal{D} \cap \mathcal{P}_S = \{y \in S \otimes \mathbb{R} \mid \forall r \in \mathrm{pr}_S(\Delta_w),\ \langle y, r_S \rangle_S \geq 0\} \cap \mathcal{P}_S.$$

Thus, the assumption that $\mathcal{D} \cap \mathcal{P}_S$ has non-empty interior leads to

$$\mathcal{D} \cap \mathcal{P}_{\mathcal{S}} = \Sigma_S(\mathrm{pr}_S(\Delta_w)) \cap \mathcal{P}_S.$$

The last expression meets and this hypothesis meet all requirements of the definition 1 of a chamber of $\mathcal{P}_S$. When applied within the framework of $\mathcal{P}_S$, this definition indeed states that a chamber D of $\mathcal{P}_S$ has have non-empty interior can be expressed as

$$D = \Sigma_S(\Delta) \cap \mathcal{P}_S$$

for some subset $\Delta \subset \mathcal{N}_S$, where

$$\mathcal{N}_S = \{x \in S \otimes \mathbb{R} \mid \langle x, x \rangle_S < 0\}.$$

We still have to show that $\mathrm{pr}_S(\Delta_w) \subset \mathcal{N}_S$, where we recall that

$$\Delta_w = \left\{x \in \Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}) \mid (x)^{\perp} \cap \mathcal{P}_S \neq \emptyset\right\}. \tag{1.7}$$

To do so, recall that proposition 10 states that an element $x \in \mathcal{R}_{\mathbb{L}}$ satisfies $(x)^{\perp} \cap \mathcal{P}_S \neq \emptyset$ if and only if $\langle x_S, x_S \rangle_S < 0$ holds. We then immediately obtain the inclusion

$$\mathrm{pr}_S(\Delta_w) \subset \mathcal{N}_S$$

so that the set $\mathrm{pr}_S(\Delta_w)$ is a defining set of the $\mathcal{P}_S$-chamber $D = \mathcal{D} \cap \mathcal{P}_S$ with Weyl vector w. As is done in Shimada's article [19], we now let

$$\mathcal{R}_{\mathbb{L}|S} = \left\{x_S \in S \otimes \mathbb{Q} \mid x \in \mathcal{R}_{\mathbb{L}},\ x_S^2 < 0\right\}$$

and

$$\mathcal{R}_S = \left\{x \in S \mid x^2 = -2\right\}.$$

Note that the inclusion $\mathcal{R}_S \subset \mathcal{R}_{\mathbb{L}|S}$ obviously holds. Moreover, the equivalence stated in proposition 10 enables us to express $\mathcal{R}_{\mathbb{L}|S}$ as

$$\mathcal{R}_{\mathbb{L}|S} = \left\{ x_S \in S \otimes \mathbb{Q} \mid x \in \mathcal{R}_{\mathbb{L}},\ (x)^{\perp} \cap \mathcal{P}_S \neq \emptyset \right\}.$$

We then immediately see that the set $\mathrm{pr}_S(\Delta_w)$ satisfies by definition

$$\mathrm{pr}_S(\Delta_w) \subset \mathcal{R}_{\mathbb{L}|S}.$$

Let $\mathcal{D}$ be a $\mathcal{P}_{\mathbb{L}}$-chamber with Weyl vector $w \in \mathbb{L}$ and assume that $\mathcal{D} \cap \mathcal{P}_S$ has non-empty interior. We have seen that $D = \mathcal{D} \cap \mathcal{P}_{\mathcal{S}}$ can be expressed as

$$D = \Sigma_S(\mathrm{pr}_S(\Delta_w)) \cap \mathcal{P}_S$$

with Δ_w defined in expression (1.7), is a $\mathcal{P}_S$-chamber, and such that

$$\mathrm{pr}_S(\Delta_w) \subset \mathcal{R}_{\mathbb{L}|S}.$$

Our above discussion led us to the following important proposition

Proposition 12. *If $\mathcal{D}$ is a $\iota(S)$-nondegenerate $\mathcal{P}_{\mathbb{L}}$-chamber with Weyl vector w then the set $pr_S(\Delta_w)$ is a defining set of the induced $\mathcal{P}_S$-chamber $D = \mathcal{D} \cap \mathcal{P}_S$.*

Given an element $\mathcal{R}_{\mathbb{L}|S}$, we define

$$(v)^{\perp} = \{ x \in S \otimes \mathbb{R} \mid \langle x, v \rangle_S = 0 \} \cap \mathcal{P}_S.$$

In his article [19, section 5], Shimada established that the collection

$$\left\{ (v)^{\perp} \mid v \in \mathcal{R}_{\mathbb{L}|S} \right\}$$

is locally finite. It is thus clear that the $\mathcal{P}_S$-chambers, which are by definition induced by $\mathcal{P}_{\mathbb{L}}$-chambers, belong to the chamber structure on $\mathcal{P}_S$ obtained by

taking the closure of connected components of

$$\mathcal{P}_S \setminus \bigcup_{v \in \mathcal{R}_{\mathbb{L}|S}} (v)^{\perp}.$$

This chamber structure will be referred to as the $\mathcal{P}_S$-chamber structure, or as the induced chamber structure. What we discussed is summarized in the following result:

Proposition 13. *Assume that $\mathcal{C}_{\mathcal{R}_{\mathbb{L}}}$ is a chamber structure on $\mathcal{P}_{\mathbb{L}}$ and that S is primitively embedded into $\mathbb{L}$ in such a way that $\mathcal{P}_S \subset \mathcal{P}_{\mathbb{L}}$. Then the collection*

$$\mathcal{C}_{\mathcal{R}_{\mathbb{L}|S}} := \{\mathcal{D} \cap \mathcal{P}_S \mid \mathcal{D} \in \mathcal{C}_{\mathbb{L}}, \exists U \subset \mathcal{P}_S, U \neq \emptyset, U \text{ open s.t. } U \subset \mathcal{D} \cap \mathcal{P}_S\}$$

is a chamber structure on $\mathcal{P}_S$ induced by the chamber structure $\mathcal{C}_{\mathbb{L}}$ on $\mathcal{P}_{\mathbb{L}}$.

An important fact regarding defining sets of induced chamber is provided by [19, Proposition 5.7]:

Proposition 14. *For any Weyl vector $w \in \mathbb{L}$, the set Δ_w is finite. In particular, any $\mathcal{R}_{\mathbb{L}|S}$-chamber D has a finite defining set.*

Recall that we denote by ι the embedding $\iota : S \hookrightarrow \mathbb{L}$ which is assumed to embed S primitively into an even hyperbolic lattice $\mathbb{L}$ chosen according to the table provided at the beginning of this section. The following definition characterizes $\mathcal{P}_{\mathbb{L}}$-chambers inducing chambers on $\mathcal{P}_S$.

Definition 15. A $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ having such that the intersection $\mathcal{D} \cap \mathcal{P}_S$ has non-empty interior is said to be $\iota(S)$-*nondegenerate*.

Please keep in mind that the $\iota(S)$-nondegeneracy is a property that depends on the transformation used to embed S into $\mathbb{L}$. We use the prefix $\iota(S)$ to emphasize this fact. Note that the classical theory, built by Shimada, instead uses the prefix S, thus neglecting to highlight the dependence of the notion of nondegeneracy on an embedding. It should be noted that Shimada provides in [19, Criterion 5.7] the following helpful criterion to check whether a $\mathcal{P}_{\mathbb{L}}$-chamber is

$\iota(S)$-nondegenerate.

Proposition 16. *A $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ with Weyl vector w is $\iota(S)$-nondegenerate if and only if there exists an element $v \in \mathcal{P}_S$ satisfying the finite number of inequalities*

$$\langle v, x\rangle_S > 0 \quad \textit{for any } x \in \textit{pr}_S(\Delta_w).$$

This criterion makes perfect sense: Let $\mathcal{D}$ be a $\iota(S)$-nondegenerate $\mathcal{P}_S$-chamber with Weyl vector w. By definition, the intersection $D = \mathcal{D} \cap \mathcal{P}_S$ has non-empty interior. That is, there exists an element $v \in \mathcal{P}_S$ such that $v \in \text{Int}(\mathcal{D} \cap \mathcal{P}_S)$. Such an element must satisfy

$$\langle v, q\rangle_S > 0$$

for all $q \in \Omega(D)$, the set of walls of D. Since we have seen in proposition 12 that $\text{pr}_S(\Delta_w)$ is a defining set of D, we have $\Omega(D) \subseteq \text{pr}_S(\Delta_w)$ by what we have seen in 1.1.2. Thus, if the above inequalities hold for all $q \in \text{pr}_S(\Delta_w)$, they also hold for all $q \in \Omega(D)$. Proposition 14 then guarantees the finiteness of $\text{pr}_S(\Delta_w)$. Thus, there are only a finite number inequalities to be checked. Our implementation of this criterion is the procedure **DegenTest**, which takes as input the data of the set of $\text{pr}_S(\Delta_w)$ associated with a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ with Weyl vector $w \in \mathbb{L}$, the data of an ample class $a_0 \in \mathcal{P}_S$, and determines whether the inequalities mentioned above all hold.

We conclude this section with an important remark: By abuse of language, it is customary to say that the Weyl vector $w \in \mathbb{L}$ of a $\iota(S)$-nondegenerate $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ is also the Weyl vector of the $\mathcal{P}_S$-chamber

$$D = \mathcal{D} \cap \mathcal{P}_S$$

it induces. The scope of the definition 11 of a Weyl vector is thus extended by inheritance to induced chambers.

1.3 Toolbox

Recall that an integral lattice such as the Néron-Severi group S of a $K3$ surface X is a sublattice of its dual $S^\vee$, defined as

$$S^\vee = \{x \in S \otimes \mathbb{R} \mid \forall y \in S,\ \langle x, y\rangle \in \mathbb{Z}\}\,.$$

We recall that S is assumed to be primitively embedded into one of the three even hyperbolic lattices $\mathbb{L}$ displayed in the table presented at the beginning of the previous section. For convenience, most computations in our programs involving S or its orthogonal complement R, both viewed as sublattices of $\mathbb{L}$, are carried out within the framework of their respective duals $S^\vee$ and $R^\vee$. No matter if we had to calculate sublattices, duals, Gram matrices, orthogonal complements, kernels, it is clear that our extensive use of functions from libraries such as the SageMath library or the SciPy library enabled us to do whatever we wanted without restriction. However, we think that we should still explain the basics mechanics behind these lattice-related functions. We already mentioned numerous times that there are three possible lattices which can be used as ambient lattices depending on the Picard number of X. We detail some basic mechanics in the framework of the ambient lattice $\mathbb{L} = U \oplus E_8(-1)$ which has rank 10, the smallest rank among the three, so that all the techniques demonstrated in this section can be applied to the two other lattices of rank 18 and 26 since $U \oplus E_8(-1)$ is naturally embedded into them. Denote by $E_8(-1)$ the integral lattice for which a Gram matrix is

$$\begin{pmatrix}
-2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -2
\end{pmatrix}$$

and denote by U the integral lattice for which a Gram matrix is

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

These two matrices enable us to obtain a Gram matrix for the direct sum lattice

$$\mathbb{L} = U \oplus E_8(-1)$$

in the obvious way.

Note that Shimada uses a basis u_1, u_2 for U in his article [19] which yields the Gram matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix}$$

for this lattice. The change of basis

$$u_1 \longmapsto u_1$$

$$u_2 \longmapsto u_2 - u_1$$

enables us to obtain the Gram matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

for U which will be used during this thesis. Please bear in mind that we thus applied the transformation mentioned above to all the results and formulas provided in Shimada's article in order to make things work with our standard basis for U. As shown in this online example, our programs can nevertheless handle input data containing embedding vectors with U-coordinates expressed in terms of the basis for U used by Shimada. Assume that u_1, u_2 form a basis for U such that the above Gram matrix for this lattice is obtained, and assume given elements $e_1, \ldots, e_8$ forming a basis for $E_8(-1)$ in such a way that the latter has

the matrix mentioned at the beginning of this section as Gram matrix. As is usually done, the direct sum

$$\mathbb{L} = U \oplus E_8(-1)$$

is endowed with the concatenated basis

$$\{u_1, u_2, e_1, \ldots, e_8\}.$$

Let X be a complex $K3$ surface S of Picard number $\rho_X < 10$ and assume that $\beta_1, \ldots, \beta_\rho$ form a basis of its Néron-Severi group S with Gram matrix G_S. We use the notation

$$[\alpha_1, \alpha_2, \ldots, \alpha_\rho]_S$$

to denote the coordinates of an element

$$D = \alpha_1 b_1 + \alpha_2 b_2 + \cdots + \alpha_\rho b_\rho$$

expressed in terms of the basis $b_1, \ldots, b_\rho$ for S. We now assume that S is primitively embedded into the even hyperbolic lattice

$$\mathbb{L} = U \oplus E_8(-1)$$

that is, we assume that there is a primitive embedding of lattices

$$\iota : S \hookrightarrow \mathbb{L}$$

defined by

$$\iota : \alpha_1 b_1 + \alpha_2 b_2 + \cdots + \alpha_\rho b_\rho \longmapsto \alpha_1 s_1 + \alpha_2 s_2 + \cdots + \alpha_\rho s_\rho$$

where $s_1, \ldots, s_\rho \in \mathbb{L}$ denote the basis vectors of S viewed as a sublattice of $\mathbb{L}$.

That is, we have

$$\iota(b_i) = s_i, \qquad 1 \leq i \leq 10$$

so that S will be identified with its image in $\mathbb{L}$ until the very end of this section, and ι can be viewed as an inclusion map of S into $\mathbb{L}$. As an immediate consequence of the fact that we are here dealing with an embedding of lattices, the Gram matrix G_S of S is preserved. Denote by $R = S^{\perp}$ be the orthogonal complement of S in $\mathbb{L}$ and denote by $r_1, \ldots, r_{10-\rho} \in \mathbb{L}$ elements forming a basis of the lattice R viewed as a sublattice of $\mathbb{L}$.

Embeddings: Since S is assumed to be primitively embedded into $\mathbb{L}$, expressing elements of S in terms of the basis of $\mathbb{L}$ is an important operation. Denote by E_S be the $(\rho \times 10)$-sized matrix with rows $s_1, \ldots, s_\rho$. The transformation

$$x \longmapsto xE_S$$

associated with E_S enables us to view this matrix as the matrix associated with the primive embedding

$$\iota : S \hookrightarrow \mathbb{L}$$

of S into $\mathbb{L}$. Let E_R be the $((10-\rho) \times 10)$-sized matrix with rows $r_1, \ldots, r_{10-\rho}$. Then the transformation

$$x \in R \longmapsto xE_R \in \mathbb{L}$$

defines an embedding of R into $\mathbb{L}$. Denote by $G_{\mathbb{L}}$ the Gram matrix of the lattice $\mathbb{L}$, and denote by G_S and G_R the respective Gram matrices of the lattices S and R. As indicated at the beginning of this section, we recall that the dual of a lattice L is the set

$$L^{\vee} = \{x \in L \otimes \mathbb{R} \mid \forall y \in L,\ \langle x, y \rangle_L \in \mathbb{Z}\}$$

and note that an integral lattice is always contained in its dual:

$$L \subseteq L^\vee.$$

We hence denote by $S^\vee$ the dual of S and denote by $R^\vee$ the dual of R. Note that $S^\vee$ is a free module of rank ρ over the integers, and if we see it as a submodule of $\mathbb{L} \otimes \mathbb{Q}$ it is then spanned by the rows of the matrix $G_S^{-1} E_S$ and denote by

$$s_1^\vee, \ldots, s_\rho^\vee \in \mathbb{L} \otimes \mathbb{Q}$$

the basis vectors of $S^\vee$ obtained from the rows of this matrix. Similarly, note that $R^\vee$ can be viewed as a free submodule of $\mathbb{L} \otimes \mathbb{Q}$ of rank $10 - \rho$ over the integers spanned by the rows of the matrix $G_R^{-1} E_R$ and denote by

$$r_1^\vee, \ldots, r_{10-\rho}^\vee \in \mathbb{L} \otimes \mathbb{Q}$$

the basis vectors of $R^\vee$ obtained from the rows of this matrix. The respective Gram matrices $G_{S^\vee}$ (resp. $G_{R^\vee}$) of $S^\vee$ (resp. $R^\vee$) relative to the basis $s_1^\vee, \ldots, s_\rho^\vee$ (resp. $r_1^\vee, \ldots, r_{10-\rho}^\vee$) are given by the formulas:

$$G_{S^\vee} = G_S^{-1} E_S \, G_{\mathbb{L}} (G_S^{-1} E_S)^t \quad \text{and} \quad G_{R^\vee} = G_S^{-1} E_S \, G_{\mathbb{L}} (G_S^{-1} E_S)^t.$$

Orthogonal Projections onto S and R

We explain how to compute orthogonal projections from $\mathbb{L}$ onto S and R. Denote by

$$A = \begin{pmatrix} s_1, & \cdots & s_\rho, & r_1, & \cdots & , r_{10-\rho} \end{pmatrix}$$

be the (10×10)-sized matrix whose columns are taken to be the basis vectors of the lattices S and R. The matrix

$$P = (A^{-1})^T$$

is used to define a transformation

$$\mathbb{L} \otimes \mathbb{Q} \longrightarrow (S \otimes \mathbb{Q}) \oplus (R \otimes \mathbb{Q})$$

defined by

$$x \longmapsto xP = \left[x_S^{(1)}, \ldots, x_S^{(\rho)}, x_R^{(1)}, \ldots, x_R^{(10-\rho)}\right]$$

which enables us to obtain the coordinates of an element $x \in \mathbb{L} \otimes \mathbb{Q}$ with respect to the basis of $(S \otimes \mathbb{Q}) \oplus (R \otimes \mathbb{Q})$. The latter is obtained by noting that we have

$$\mathbb{L} \subset \mathbb{L} \otimes \mathbb{Q},$$

so that the basis $\{s_1, \ldots, s_\rho\}$ for S can be viewed as a basis of

$$S \otimes \mathbb{Q} \subset \mathbb{L} \otimes \mathbb{Q}.$$

Similarly, the basis $\{r_1, \ldots, r_{10-\rho}\}$ for S can be viewed as a basis of

$$R \otimes \mathbb{Q} \subset \mathbb{L} \otimes \mathbb{Q}.$$

A basis of

$$(S \otimes \mathbb{Q}) \oplus (R \otimes \mathbb{Q})$$

can thus be obtained from the concatenated basis

$$\{s_1, \ldots, s_\rho, r_1, \ldots, r_{10-\rho}\}$$

of $S \oplus R$. We also note that there is an equality

$$\mathbb{L} \otimes \mathbb{Q} = (S \otimes \mathbb{Q}) \oplus (R \otimes \mathbb{Q}).$$

Denote by P_S be the $(10 \times \rho)$-sized matrix obtained by taking as columns the first ρ columns of the matrix P. The matrix P_S is associated with the projection transformation

$$x \in \mathbb{L} \otimes \mathbb{Q} \longmapsto xP_S \in S \otimes \mathbb{Q}$$

from $\mathbb{L} \otimes \mathbb{Q}$ onto $S \otimes \mathbb{Q}$ and enables us to obtain coordinates of projections

$$xP_S = \left[x_S^{(1)}, \dots x_S^{(\rho)}\right]$$

of elements $x \in \mathbb{L} \otimes \mathbb{Q}$ with respect to the basis of $S \otimes \mathbb{Q}$. Denote by P_R be the $(10 \times (10 - \rho))$-sized matrix obtained by extracting the columns of P ranging from the $(\rho + 1)$-th to the last one. The matrix P_R is the matrix associated with the projection tranformation

$$x \longmapsto xP_R = \left[x_R^{(1)}, \dots x_R^{(10-\rho)}\right]$$

from $\mathbb{L} \otimes \mathbb{Q}$ onto $R \otimes \mathbb{Q}$ enables us to get coordinates of projections onto R of elements $x \in \mathbb{L}$ with respect to the basis of $R \otimes \mathbb{Q}$.

Projections onto $S^\vee$ and $R^\vee$

Projections from $\mathbb{L}$ onto $S^\vee$ and $R^\vee$ are also common operations. We, however, made use of two distinct bases of $S^\vee$ to consider two ways of defining projections from $\mathbb{L} \otimes \mathbb{Q}$ into $S^\vee$.

- One basis is denoted by $\mathcal{B}_1$ and made of elements of $\mathbb{L} \otimes \mathbb{Q}$. Using this base makes sense when S and $S^\vee$ are considered within the framework of a primitive embedding of S into $\mathbb{L}$.
- The other basis is denoted by $\mathcal{B}_2$ and made of elements of $S \otimes \mathbb{Q}$. Using this base makes sense when considering S and $S^\vee$ outside of the framework of the primitive embedding of S into $\mathbb{L}$.

Projections onto $S^\vee$ can indeed be either considered within the framework of the embedding of S into $\mathbb{L}$ or by viewing S as a lattice of its own right. The first approach is especially convenient when using SageMath, whose lattice features enable us to easily define S and R as sublattices of $\mathbb{L}$ and thus perform all computations in this framework. A basis $\mathcal{B}_1$ for $S^\vee$ is obtained by taking as elements the rows of the matrix $G_S E_S$. Note that this basis will be used by Sage

for $S^\vee$ whenever S is defined as a sublattice of $\mathbb{L}$. Consider the (10×10)-sized matrix

$$B = \begin{pmatrix} s_1^\vee & , \cdots & , s_\rho^\vee, & r_1^\vee & , \cdots , & r_{10-\rho}^\vee \end{pmatrix}$$

obtained by taking as columns the elements of the concatenated basis

$$\left\{ s_1^\vee, \ldots, s_\rho^\vee, r_1^\vee, \ldots, r_{10-\rho}^\vee \right\}$$

of the direct sum $S^\vee \oplus R^\vee$. The matrix

$$Q := (B^{-1})^t$$

then yields a transformation

$$\mathbb{L} \otimes \mathbb{Q} \longrightarrow S^\vee \oplus R^\vee$$

defined by

$$x \longmapsto xQ = \left[x_{S^\vee}^{(1)}, \ldots, x_{S^\vee}^{(\rho)}, x_{R^\vee}^{(1)}, \ldots, x_{R^\vee}^{(10-\rho)} \right]$$

which the coordinates of elements $x \in \mathbb{L} \otimes \mathbb{Q}$ with respect to the above-mentioned basis concatenated basis for $S^\vee \oplus R^\vee$. Obtaining the coordinates of the projections onto $S^\vee$ and $R^\vee$ is easy:

- $x_{S^\vee}^{(1)}, \ldots, x_{S^\vee}^{(\rho)}$ are the coordinates of the projection of x into $S^\vee$.
- $x_{R^\vee}^{(1)}, \ldots, x_{R^\vee}^{(10-\rho)}$ are the coordinates of the projection of x into $R^\vee$.

These coordinates can also be obtained by proceeding as follows: If we let $Q_{S^\vee}^{\mathcal{B}_2}$ be the $(10 \times \rho)$-sized matrix with columns obtained by extracting the first ρ columns of the matrix Q, then a projection

$$\mathrm{pr}_{S^\vee}^{\mathcal{B}_2} : \mathbb{L} \otimes \mathbb{Q} \longrightarrow S^\vee$$

defined by

$$\mathrm{pr}_{S^\vee}^{\mathcal{B}_2} : x \longmapsto xQ_{S^\vee}^{\mathcal{B}_1} = \left[x_{S^\vee}^{(1)}, \dots x_{S^\vee}^{(\rho)}\right]$$

is obtained. We can also consider projections into $S^\vee$ endowed with basis

$$\mathcal{B}_2 = \left\{\mathrm{col}_1(G_S^{-1}), \dots, \mathrm{col}_\rho(G_S^{-1})\right\}$$

obtained by taking the columns of the inverse G_S^{-1} of the Gram matrix of S. Doing so amounts to considering S as a lattice of its own, and not as a primitive sublattice of $\mathbb{L}$. This approach is convenient for computations that occur within procedures that produce transformations of $O(S)$, that is, within the procedures **CongChecker** and **AutChamber** described in sections 1.7.4 and 1.7.3. A projection operator

$$\mathrm{pr}_{S^\vee}^{\mathcal{B}_2} : \mathbb{L} \otimes \mathbb{Q} \longmapsto S^\vee$$

from $\mathbb{L} \otimes \mathbb{Q}$ onto $S^\vee$ endowed with its basis $\mathcal{B}_2$ is obtained by defining

$$\mathrm{pr}_{S^\vee}^{\mathcal{B}_2} : x \longmapsto xG_{\mathbb{L}}E_S^T.$$

Considering distinct bases as done for $S^\vee$ would make no sense in the case of $R^\vee$. Indeed, recall that R is defined as the orthogonal complement of S into $\mathbb{L}$. We therefore have no other choice but to take a basis of $R^\vee$ within the framework of the embedding. Such a basis is obtained by taking as basis elements the rows of the matrix $G_R E_R$. Define $Q_{R^\vee}$ as the $10 \times (10 - \rho)$ matrix whose columns are obtained by extracting the last $10 - \rho$ columns columns of the matrix Q. A projection

$$\mathrm{pr}_{R^\vee}^{\mathcal{B}_2} : \mathbb{L} \otimes \mathbb{Q} \longrightarrow R^\vee$$

defined by

$$\mathrm{pr}_{R^\vee} : x \longmapsto xQ_{R^\vee} = \left[x_{R^\vee}^{(1)}, \dots, x_{R^\vee}^{(10-\rho)}\right]$$

is then obtained.

Embeddings of $S^\vee$ and $R^\vee$ into $\mathbb{L}\otimes\mathbb{R}$

Denote by

$$E_{S^\vee} = G_S^{-1}E_S$$

the basis matrix of $S^\vee$. Note that the rows of this matrix are the elements of $\mathcal{B}_2$. Let $v_S \in S^\vee$ with coordinates expressed with respect to this basis. We denote by $v_S^{\mathbb{L}\otimes\mathbb{Q}}$ the image of $v_S \in S^\vee$ under the map

$$v_S \longmapsto v_S^{\mathbb{L}\otimes\mathbb{Q}} = v_S E_{S^\vee}$$

from $S^\vee$ into $\mathbb{L}\otimes\mathbb{Q}$. Analogously, let

$$E_{R^\vee} = G_R^{-1}E_R$$

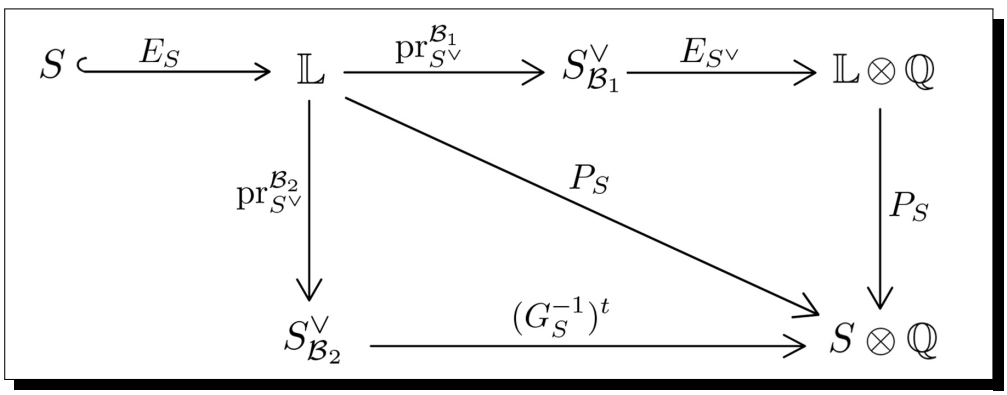
be the basis matrix of $R^\vee$. Consider an element $v_R \in R^\vee\otimes\mathbb{Q}$ with coordinates expressed with respect to the basis obtained by taking the rows of this matrix. We denote by $v_R^{\mathbb{L}\otimes\mathbb{Q}}$ the image of v_R under the map

$$v_R \in R^\vee \longmapsto v_R^{\mathbb{L}\otimes\mathbb{Q}} = v_R E_{R^\vee}$$

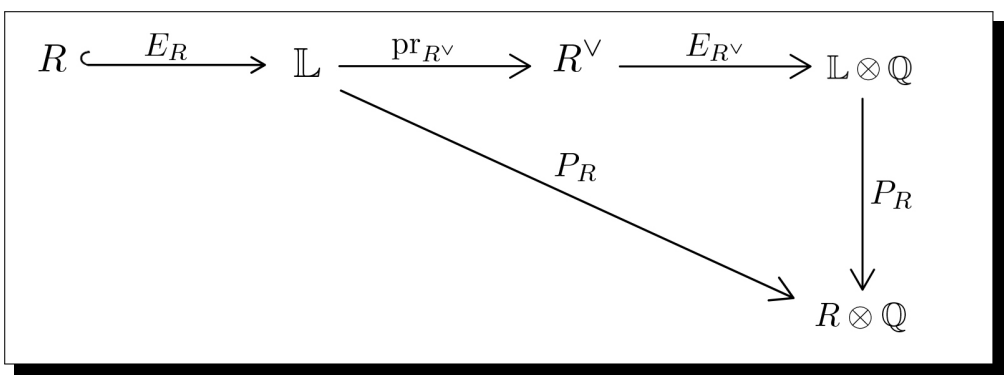
from $R^\vee$ into $\mathbb{L}\otimes\mathbb{Q}$. The two following figures summarize the material discussed in this section: We embed S into $\mathbb{L}$ by right multiplication by the matrix E_S. We can consider projections into $S^\vee$ by either regarding it endowed with its basis $\mathcal{B}_1$ which is denoted by $S^\vee_{\mathcal{B}_1}$, or with its basis $\mathcal{B}_2$ which is denoted by $S^\vee_{\mathcal{B}_2}$.

From the framework of the basis $S^\vee_{\mathcal{B}_1}$, projection into $S\otimes\mathbb{Q}$ requires going back into $\mathbb{L}\otimes\mathbb{Q}$ by right multiplication by $E_{S^\vee}$ and then project into $S\otimes\mathbb{Q}$ by right multiplication by P_S. Projection into $S\otimes\mathbb{Q}$ requires left multiplication by $(G_S^{-1})^t$ when working in the framework of $S^\vee_{\mathcal{B}_2}$.

The following figures summarize all the material discussed in this section :



The mechanics are similar regarding R and its dual $R^\vee$, except that we only consider a single basis made of vectors of $\mathbb{L}\otimes\mathbb{Q}$ for the latter.



1.4 Shimada's enhanced Short Lattice Vectors Enumerator

Among the numerous features of popular computer algebra systems (CAS) can be found short lattice vectors enumeration functions. Given a positive definite gram matrix Q of a rank n lattice $\mathcal{L}$ and an integer c as input data, a short lattice vectors function returns the set of all lattice elements $x \in \mathcal{L}$ satisfying

$$xQx^T \leq c. \tag{1.8}$$

As far as we know, there is no CAS (in 2022) that integrates a function capable of determining the solution set an expression of the form

$$xQx^T + 2xL \leq c, \tag{1.9}$$

where L is an n-sized column vector. In his article [18, Section 3.1], Shimada provides an algorithm to determine the solution set of an expression such as (1.9). We used the SageMath Python library in order to produce an implementation of this algorithm. The result is the function **ShiVectors**, detailed and available for download on K3surfaces.com. In this section, we build on the structure outlined by Shimada in his article [18, Section 3.1] and introduce this key algorithm from a purely pragmatic point of view. Our goal consists in providing guidelines so that the readers can easily implement this algorithm.

Definition 17. A quadratic triple of n variables is a triple $[Q, L, c]$ where Q is a $(n \times n)$-sized symmetric matrix with rational entries, where L is a column vector of length n with rational entries, and where c is a rational number.

When the matrix Q is positive definite, the triple

$$[Q, L, c]$$

is called a positive quadratic triple. When Q is negative definite, we bring the problem back to the positive definite case by substituting $-Q$ to Q.

To any triple $QT = [Q, L, c]$ can be associated a quadratic function

$$q_{QT} : \mathbb{Q}^n \longrightarrow \mathbb{Q}$$

defined by

$$q_{QT}(\boldsymbol{x}) := \boldsymbol{x}Q\boldsymbol{x}^t + 2\boldsymbol{x}L + c.$$

The remainder of this section will be based on the section 3.1 "*An algorithm for a positive quadratic triple*" from Shimada's article [18]. We provide the necessary details and clarifications which will enable the readers to easily produce their own implementations of Shimada's algorithm to compute the set

$$E(QT) = \{x \in \mathbb{Z}^n \mid q_{QT}(x) \leq 0\}.$$

Also, please remember that we provide our own ready-to-use implementation of this algorithm, which is called **ShiVectors**, on k3surfaces.com The main routine used in Shimada's algorithm consists in applying sequences of projection operations. The purpose of a projection consists in returning a triple of $m-1$ variables from the input of a triple of m variable. By repeated applications of projections, we finally obtain a triple of a single variable. The degree 1 equation associated with this triple has a solution set that can be determined without hassle. Let $QT = [Q, L, c]$ be a positive quadratic triple of n variables.

Projection procedure n°1:

Following Shimada's guidelines, we arrange the elements of this triple as follows:

$$Q = \left[\begin{array}{c|c} Q' & \boldsymbol{p}' \\ \hline {}^t\boldsymbol{p}' & r' \end{array}\right], \qquad L = \left[\begin{array}{c} L' \\ \hline m' \end{array}\right]$$

where Q is a $(n-1)\times(n-1)$-sized square matrice, where p' and L' are column vectors of length $n-1$, and where r' and m' are rationals.

Since the matrix Q is assumed to be positive definite, note that $r' > 0$. Shimada states that a quadratic triple of $n-1$ variables is then obtained from the triple QT by the formula

$$\mathbf{pr}(QT) := \left[Q' - \frac{1}{r'}(\boldsymbol{p'}\,{}^t\boldsymbol{p'}), L' - \frac{m'}{r'}\boldsymbol{p'}, c - \frac{m'^2}{r'}\right].$$

Projection procedure n°2:

We follow Shimada's guidelines and arrange the elements of the triple QT as follows:

$$Q = \left[\begin{array}{c|c} r'' & {}^t\boldsymbol{p}'' \\ \hline \boldsymbol{p}'' & Q'' \end{array}\right], \qquad L = \left[\begin{array}{c} m'' \\ \hline L'' \end{array}\right]$$

where Q'' is a $(n-1)\times(n-1)$-sized square matrix , where p'' and L'' are column vectors of length $n-1$, and where $r''$and m'' are rationals. As before, we note that $r'' > 0$ due to the assumed positive definiteness of the matrix Q. Let $a \in \mathbb{Q}$ be a rational number. Shimada states that a quadratic triple of $n-1$ variables $\iota^*(a, QT)$ is then obtained by the formula:

$$\iota^*(a, QT) := \left[Q'', a\boldsymbol{p''} + L'', a^2r'' + 2am'' + c\right]. \tag{1.10}$$

This procedure can be executed more than one time, say $m < n$ times, as follows. Let

$$\boldsymbol{a} = [a_1, \ldots, a_m] \in \mathbb{Q}^m.$$

A positive quadratic triple $\iota^*(\boldsymbol{a}, QT)$ of $(n-m)$-variables is then obtained by m

sucessive applications of the formula given in expression (1.10). That is, define

$$QT^0 := QT, \qquad QT^{\nu+1} := \iota^*(a_{\nu+1}, QT^\nu), \qquad \iota^*(\boldsymbol{a}, QT) := QT^m$$

where $\nu = 0, \ldots, m-1$.

1.4.1 ShiVectors - Our implementation of Shimada's SLVE

Assume that an initial positive quadratic triple

$$QT_n^0 := QT$$

of n-variables is given.

By $n-1$ applications the projection procedure n°2 described above, compute

$$QT_{i-1}^0 = \mathrm{pr}(QT_i^0)$$

for $2 \leq i \leq n$ and note that

$$QT_1^0 = \mathrm{pr}(QT_2^0)$$

is a triple of a single variable, whose associated degree 1 equation has a solution set which can be easily determined. Denote by $\mathscr{S}(QT)$ the set containing these triples:

$$\mathscr{S}(QT) = \left\{QT_1^0, QT_2^0, \ldots, QT_n^0\right\}.$$

Assume given an initial positive quadratic triple QT of n variables. We now state the main procedure behind Shimada's algorithm to compute $E(QT)$. First, we warn the reader that the following procedure is recursive.

Procedure ShiVectors : The input consists of a parameter $\nu \in \mathbb{Z}$, of sets $\mathcal{Z}$, $\mathcal{E}$ and $\mathcal{Y}$, so that the procedure can be formalized as

$$\textbf{ShiVectors}(\nu, \mathcal{Z}, \mathcal{Y})$$

Note that the sets $\mathcal{Z}$ and $\mathcal{E}$ will initially be taken as empty sets in order to initiate the procedure, while $\mathcal{Y}$ will be initially taken as $\mathscr{S}(QT)$ and is thus assumed to be a set of triples, as explained earlier. The first thing that the procedure does consists in taking a look at the value of the parameter ν:

- If $\nu = n + 1$, append the list $\mathcal{Z}$ to the list $\mathcal{E}$, return the list $\mathcal{E}$ as output, end of story.
- Otherwise, denote by $\mathcal{X}(\mathscr{S}(QT))$ the *solution* set of the inequality obtained from the triple a single variable QT_1^0 contained in $\mathscr{S}(QT)$, and proceed as follows.

Denote by $\mathscr{S}^*(QT)$ a copy of the set $\mathscr{S}(QT)$ from which the triple of a single variable QT_1^0 has been removed. For each $q \in \mathcal{X}(\mathscr{S}(QT))$, Shimada instructs to proceed as follows:

(i) Create a copy $\mathcal{Z}'$ of the set $\mathcal{Z}$ and compute the set

$$\mathscr{S}_{\text{update}}(QT) := \{\iota^*(q, r) \mid r \in \mathscr{S}^*(QT)\}\,.$$

(ii) Append q to the list $\mathcal{Z}'$ and execute $\textbf{ShiVectors}(\nu + 1, \mathcal{Z}', \mathscr{S}_{\textbf{update}}(QT), \mathcal{E})$.

That is all what is to be done. In practice, given a triple

$$QT = [Q, L, c]$$

we execute the procedure **ShiVectors** with

$$\nu = 1, \quad \mathcal{Z} = \{\,\}\,.$$

$$\textbf{ShiVectors}(1, \mathcal{Z} = \{\}, \mathscr{S}(QT), \mathcal{E} = \{\})$$

Shimada then guarantees that the set returned by this procedure is

$$E(QT) = \{x \in \mathbb{Z}^n \mid q_{QT}(x) \leq 0\},$$

as desired.

1.4.2 Applications - ShiChecker & ShiBooster

The two following algorithms due to Shimada are applications of **ShiVectors**. Note that additional details on these applications can be found in [18] and the second part of this thesis.

Procedure ShiChecker: Let L be a hyperbolic lattice, let v be a vector of $L \otimes \mathbb{Q}$ satisfying $v^2 > 0$, let α be a rational number, and let d be an integer. The finite set

$$\{x \in L \mid \langle x, v\rangle_L = \alpha,\ \langle x, x\rangle_L = d\}$$

can be computed by the method decribed in [18, Section 3.2].

Procedure ShiBooster: Let L be a hyperbolic lattice, let v, h be vectors of $L \otimes \mathbb{Q}$ such that

$$\langle v, h\rangle_L > 0,\ \langle h, h\rangle_L > 0,\ \langle v, v\rangle_L > 0,$$

and let d be a negative integer. Then the finite set

$$\{x \in L \mid \langle v, x\rangle_L < 0,\ \langle h, x\rangle_L > 0,\ \langle x, x\rangle_L = d\}$$

can be computed by the method described in [18, Section 3.3].

Our implementations of these algorithms due to Shimada are available for download on K3surfaces.com as **ShiBooster** and **ShiChecker**, respectively.

1.5 Computing the walls of an induced chamber

We have seen in section 1.1.2 that a $\iota(S)$-nondegenerate $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ induces a $\mathcal{P}_S$-chamber

$$D = \mathcal{D} \cap \mathcal{P}_S.$$

Assuming that $\mathcal{D}$ has Weyl vector w, which is inherited by the induced chamber D, we have seen in proposition 12 that the set $\mathrm{pr}_S(\Delta_w)$ is a defining set of the induced $\mathcal{P}_S$-chamber $D = \mathcal{D} \cap \mathcal{P}_S$. The aim of this section consists in providing the procedures which will enable the reader to compute a primitively minimal defining set of an induced $\mathcal{P}_S$-chamber D from the sole input data of its Weyl vector. That is, we provide procedures to compute the data of the walls of a $\mathcal{P}_S$-chamber D. To do so, we proceed in two stages:

- In section 1.5.1, we present the procedure **DeltaW**. This procedure is based on Shimada's algorithm 5.8 from [19] and ouputs Δ_w from the input data of the Weyl vector w of a $\mathcal{P}_S$-chamber D.

- In section 1.5.2, we introduce the procedure **SetOfWalls**. The latter is based on Shimada's algorithm 3.17 from his article [19] and outputs a primitively minimal defining set from the input data of a defining set of a $\mathcal{P}_S$-chamber.

The computation of the set of walls of a $\mathcal{P}_S$-chamber $D = \mathcal{D} \cap \mathcal{P}_S$ with Weyl vector w can then be performed by proceeding as follows: Using the Weyl vector w of D as input, we use the procedure **DeltaW** to compute the set Δ_w. By proposition 12, the set $\mathrm{pr}_S(\Delta_w)$ is a defining set of D. We then apply the procedure **SetOfWalls** to the latter in order to obtain a primitively minimal defining set of D, i.e., the data of the walls of D.

1.5.1 Procedure DeltaW

As before, we work with a complex $K3$ surface X. We assume that its Néron-Severi group $S = \mathrm{NS}(X)$ has been primitively embedded into a suitable ambient

even hyperbolic lattice $\mathbb{L}$ by an embedding

$$\iota : S \hookrightarrow \mathbb{L}$$

in such a way that $\mathcal{P}_S \subset \mathcal{P}_{\mathbb{L}}$. Assume that a Weyl vector $w \in \mathbb{L}$ of a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ is given. Let $R = S^{\perp}$, the orthogonal complement of S in $\mathbb{L}$. We follow the structure outlined by Shimada in Algorithm 5.8 from his article [19] and provide all the necessary additional details which will enable our readers to produce their own implementations of this algorithm without hassle. We also provide our implementation of this algorithm, called **DeltaW**, available for download and explained on k3surfaces.com We also explain on this website how to compute R, $R^{\vee}$, G_R, n_R... and all the entities mentioned in this section using the SageMath library. We now state the algorithm provided by Shimada in [19, algorithm 5.8] and then explain how we implemented it. Assume that the Weyl vector $w \in \mathbb{L}$ of an $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ is given as input data. The following algorithm returns the set

$$\begin{aligned}\Delta_w &= \left\{x \in \Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}) \mid (x)^{\perp} \cap \mathcal{P}_S \neq \emptyset\right\} \\ &= \left\{x \in \Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}) \mid x_S^2 < 0\right\} \qquad \text{(where we used proposition 10)}\end{aligned}$$

from the input data of w. Shimada**'s Algorithm 5.8:**

- **Step n°1 -** Compute $w_S = \mathrm{pr}_{S^{\vee}}(w) \in S^{\vee}$, $w_R = \mathrm{pr}_{R^{\vee}}(w) \in R^{\vee}$ (see details in section 1.3).
- **Step n°2:** Compute the set

  $$n_R = \left\{c \in \mathbb{Q} \mid d_R c \in \mathbb{Z}, d_R^2 c \in 2\mathbb{Z}, -2 < c \leq 0\right\}$$

  where d_R denotes the order of the discriminant group $R^{\vee}/R$ of R.

  Define $\Delta' := \{\}$.
- **Step n°3 -** Let $\beta_{\max} = \max\{|\beta| \mid \beta \in n_R\}$. Use a short lattice vectors

enumeration solution to compute

$$\left\{v \in R^\vee \mid v^2 \leq \beta_{\max}\right\}$$

and process the data of this set to obtain

$$R^\vee[\beta] = \left\{v \in R^\vee \mid v^2 = \beta\right\} \quad \text{and} \quad a_R[\beta] := \{\langle w_R, v\rangle_{R^\vee} \mid v \in R^\vee[\beta]\}$$

for each $\beta \in n_R$.

- **Step n°4 -** For each pair $(\beta, \alpha) \in n_R \times a_R[\beta]$, use algorithm **ShiChecker** to compute the finite set

$$S^\vee[\beta, \alpha] = \{v \in S^\vee \mid \langle v, w_S\rangle_{S^\vee} = 1 - \alpha,\ \langle v, v\rangle_{S^\vee} = -2 - \beta\}.$$

- **Step n°5 -** For each $\beta \in n_R$, each $v_R \in R^\vee[\beta]$, each $\alpha \in a_R^\vee[\beta]$ and each $v_S \in S^\vee[\beta, \alpha]$, determine whether the element $v_S + v_R$ belongs to $\mathbb{L}$. That is, determine whether the coordinates of $v_S + v_R$ with respect to the standard basis of $\mathbb{L}$ are all integers.

  If the answer is positive, append $v_S + v_R$ to Δ'.

- **Final step:** Output Δ' as Δ_w.

Before explaining this algorithm step-by-step, we have to shed light on the general idea behind Shimada's algorithm 5.8. The endgame consists in obtaining elements of Δ_w as sums $v_S + v_R$ of elements $v_S \in S^\vee$ and $v_R \in R^\vee$ which satisfy

$$\langle v_S, v_S\rangle_{S^\vee} = -2 - \langle v_R, v_R\rangle_{R^\vee} \qquad \text{and} \qquad \langle v_S, w_S\rangle_{S^\vee} = 1 - \langle v_R, w_R\rangle_{R^\vee}.$$

To this end, **Step n°3** will be used to obtain suitable elements $v_R \in R^\vee$, while **Step n°4** will enable us to determine elements $v_S \in S^\vee$ for which there exist an element v_R such that $v_S + v_R$ satisfies the above equalities. Once this is done, we

will not be far from obtaining the set Δ_w. Indeed, assume that elements $v_S \in S^\vee$ and $v_R \in R^\vee$ satisfying the above equalities have been obtained, that is, such that equalities are given, i.e., satisfy

$$\langle v_S + v_R, v_S + v_R \rangle_{\mathbb{L}} = -2 \qquad \text{and} \qquad \langle v_S + v_R, w \rangle_{\mathbb{L}} = 1. \tag{1.11}$$

Assume furthermore that $v_S + v_R \in \mathbb{L}$, i.e., that $v_S + v_R$ has integer coordinates with respect to the standard basis of $\mathbb{L}$, and note that performing this check is the purpose of **Step n°5**.

Definition 11 states that the Weyl vector w of a $\mathcal{P}_{\mathbb{L}}$ -chamber $\mathcal{D}$ enables us to express the minimal defining set $\Delta_{\mathcal{R}_L}(\mathcal{D})$ of $\mathcal{D}$ as

$$\Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}) = \{x \in \mathbb{L} \mid \langle x, x \rangle_{\mathbb{L}} = -2,\ \langle w, x \rangle_{\mathbb{L}} = 1\},$$

thus, if $v_S + v_R$ satisfy equalities (1.11) then it is clear that

$$v_S + v_R \in \Delta_{\mathcal{R}_{\mathbb{L}}}(D).$$

Moreover, we have by definition

$$\begin{aligned} \Delta_w &= \left\{x \in \Delta_{\mathcal{R}_{\mathbb{L}}}(\mathcal{D}) \mid x_S^2 < 0\right\} \\ &= \left\{x \in L \mid \langle x, x \rangle_L = -2,\ \langle w, x \rangle_L = 1,\ \langle \mathrm{pr}_{S^\vee}(x), \mathrm{pr}_{S^\vee}(x) \rangle_{S^\vee} < 0\right\}. \end{aligned}$$

Hence, it remains to prove that the projection of $v_S + v_R$ onto $S^\vee$, which is, by definition v_S, satisfies

$$\langle v_S, v_S \rangle_{S^\vee} < 0,$$

in order to finally obtain that $v_S + v_R \in \Delta_w$. To this end, we can use the assumption $\langle v_R, v_R \rangle_{R^\vee} \in n_R$, so that

$$-2 < \langle v_R, v_R \rangle_{R^\vee} \leq 0$$

holds. The equality

$$\langle v_S, v_S \rangle_{S^\vee} + \langle v_R, v_R \rangle_{R^\vee} = -2$$

will then readily allow us to deduce

$$\langle v_S, v_S \rangle_{S^\vee} < 0.$$

Consequently, we will have finally obtained that

$$v_S + v_R \in \Delta_w$$

holds, as desired. We follow a step-by-step approach and provide all the details, tips and tricks which enabled us to successfully implement this critical algorithm due to Shimada.

Step n°1 - We start by computing the orthogonal projections of w onto $S^\vee$ and $R^\vee$, which are respectively denoted by w_S and w_R. In order to do so, we recommend to make use of the material introduced in the section 1.3 of this thesis.

Step n°2 - We compute the set n_R. First, note that the value of d_R can be obtained by computing the determinant of the Gram matrix G_R of R.

A rational $\beta \in \mathbb{Q}$ belongs to n_R if and only if there exist integers $k_1, k_2 \in \mathbb{Z}$

$$d_R\beta = k_1 \qquad \text{and} \qquad d_R^2\beta = 2k_2$$

such that

$$\begin{cases} -2d_R < k_1 \leq 0 \\ -2d_R^2 < 2k_2 \leq 0. \end{cases}$$

In order to compute n_R, define

$$A = \{k/d_R \mid k \in \mathbb{Z},\ -2d_R < k \leq 0\}$$

and

$$B = \left\{ 2k/d_R^2 \mid k \in \mathbb{Z},\ -2d_R^2 < 2k \leq 0 \right\}.$$

It is clear that there is an equality

$$n_R = A \cap B$$

and the knowledge of d_R is the only data required in order to compute A and B. We let $\Delta' = \{\,\}$.

Step n°3 - To each element $\beta \in n_R$, Shimada associates the sets

$$R^\vee[\beta] = \left\{ v \in R^\vee \mid v^2 = \beta \right\} \quad \text{and} \quad a_R[\beta] := \left\{ \langle w_R, v \rangle_{R^\vee} \mid v \in R^\vee[\beta] \right\},$$

which, as stated by Shimada in his article, are finite. Since the Gram Matrix of $R^\vee$ is negative definite, sets such as $R^\vee[\beta]$ can be easily computed using a short lattice vectors enumeration algorithm. A few tips regarding this task: First, note that nothing guarantees that the Gram matrix of $R^\vee$ has only integer entries. To be safe, we multiply $G_{R^\vee}$ by the least common multiple δ of the denominators of its entries. Also, keep in mind that we have $|\beta| < 2$ since $\beta \in n_R$. This implies that a single call for a short lattice vectors function will enable us to obtain the data of all the sets $R^\vee[\beta]$. We thus use a short lattice vectors enumerator in such a way that it returns the set

$$\left\{ x \in R^\vee \mid -x(\delta G_{R^\vee})x^T < 2\delta + 1 \right\}$$

from which all the sets $R^\vee[\beta]$ will be obtained by basic sorting. This set should not be computed every time the procedure to compute Δ_w is executed. Doing so would amount to wasting computational resources. As soon as a Gram matrix for $R^\vee$ is obtained, the above-mentioned set can be computed once and for all.

Assuming given an element $\beta \in n_R$ and computing the set $R^\vee[\beta]$ then enables

us to obtain the associated set $a_R[\beta]$ which is formed by computing

$$\langle w_R, v\rangle_{R^\vee}$$

for each $v \in R^\vee[\beta]$.

Step n°4 - Fix an element $\beta \in n_R$, an element $\alpha \in a_R[\beta]$ and an element $v_R \in R^\vee[\beta]$. That is, the equalities

$$\beta = \langle v_R, v_R\rangle_{R^\vee} \qquad \text{and} \qquad \alpha = \langle v_R, w_R\rangle_{R^\vee}$$

hold. The procedure to obtain an element $v_S \in S^\vee[\beta, \alpha]$ can be broken down into two stages:

(a) First, we determine a solution $c \in S^\vee$ of the equation $\langle x, w_S\rangle_{S^\vee} = 1 - \alpha$.

(b) We then determine an element $y \in (w_S)^\perp \subset S^\vee$ satisfying

$$\langle y + c, y + c\rangle_{S^\vee} = -2 - \beta,$$

i.e., satisfying

$$\langle y, y\rangle_{S^\vee} + 2\langle y, c\rangle_{S^\vee} + \langle c, c\rangle_{S^\vee} = -2 - \beta. \tag{1.12}$$

The element

$$v_S = y + c$$

thus assembled will then be such that

$$\begin{aligned}\langle v_S, w_S\rangle_{S^\vee} &= \langle y + c, w_S\rangle_{S^\vee} \\ &= 0 + \langle c, w_S\rangle_{S^\vee} \\ &= 1 - \alpha,\end{aligned}$$

and

$$\begin{aligned}\langle v_S, v_S\rangle_{S^\vee} &= \langle y + c, y + c\rangle_{S^\vee} \\ &= -2 - \beta,\end{aligned}$$

so that $v_S \in S^\vee[\beta, \alpha]$, as desired. Before proceeding further, we want to point out that once a basis

$$\mathcal{B} = \left\{s_1^\vee, \ldots, s_\rho^\vee\right\}$$

for $S^\vee$ is chosen, an element $x \in S^\vee$ can be expressed as

$$x = x_1 s_1^\vee + \cdots + x_\rho s_\rho^\vee$$

where $\rho = \text{rank}(S)$ and where $x_1, \ldots, x_\rho \in \mathbb{Z}$ are the coordinates of x with respect to the basis $\mathcal{B}$ of $S^\vee$. The basis $\mathcal{B}$ being implicit, the notation

$$x = [x_1, \ldots, x_\rho]$$

will be used regularly in the remainder of this section. Denote by $G_{S^\vee}$ a Gram matrix for $S^\vee$.

Implementation of (a) - First, we recall that the projection

$$w_S = \left[w_1^\vee, \ldots, w_\rho^\vee\right]$$

of the Weyl vector w onto $S^\vee$ has been computed in **Step n°1**. Remember that the section 1.3 of this thesis provides guidelines to compute projections. Solving

the equation

$$\langle x, w_S \rangle_{S^\vee} = 1 - \alpha$$

obviously amounts to determining integers $x_1, \ldots, x_\rho$ satisfying the equality

$$\begin{bmatrix} x_1 & \ldots & x_\rho \end{bmatrix} G_{S^\vee} \begin{bmatrix} w_1^\vee \\ \vdots \\ w_\rho^\vee \end{bmatrix} = 1 - \alpha. \tag{1.13}$$

The left-hand side of this expression can be arranged in such a way that (1.13) can be turned into

$$\sum_{i=1}^{\rho} \gamma_i x_i = 1 - \alpha$$

where the γ_i are elements of $\mathbb{Q}$. If necessary, clear the denominators on both sides of this expression, so that it takes the form

$$\sum_{i=1}^{\rho} \mu_i x_i - \gamma = 0 \tag{1.14}$$

where

$$\gamma \in \mathbb{Z} \qquad \text{and} \qquad \mu_i \in \mathbb{Z}$$

for $i \in \{1, \ldots, \rho\}$. A basis

$$\{\epsilon_1, \ldots, \epsilon_{\rho-1}\} \subset S^\vee$$

of the $(\rho - 1)$-dimensional solution space of the degree one equation (1.14) of the integer variables $x_1, \ldots, x_\rho$ can then be computed using a CAS.

Implementation of (b) - Before describing how we proceeded, let us provide context. The Gram matrix matrix of $S^\vee$, being indefinite, prevents us from using a short lattice vectors enumeration algorithm in order to determine the set of

elements $x \in S^\vee$ satisfying

$$\langle x, x \rangle_{S^\vee} = -2 - \beta.$$

In order to overcome this obstacle, Shimada's idea consists in determining a sublattice of $S^\vee$ on which the restriction of the bilinear form of $S^\vee$ is definite. The orthogonal complement $(w_S)^\perp$ of w_S in $S^\vee$ matches this requirement. Indeed, a result of Conway & Sloane mentioned in [19, Section 4] guarantees that a Weyl vectors $w \in \mathbb{L}$ all satisfy

$$\langle w, w \rangle_{\mathbb{L}} > 0$$

when the lattice into which S is primitively embedded is $\mathbb{L} = U \oplus E_8(-1)$ or $\mathbb{L} = U \oplus E_8(-1) + E_8(-1)$. Since R is negative definite, this implies that

$$\langle w_S, w_S \rangle_{S^\vee} > 0$$

for all Weyl vectors in the framework of these two lattices. The Hodge Index theorem then ensures that the restriction of $\langle\ ,\ \rangle_{S^\vee}$ to $(w_S)^\perp$ is negative definite, hence enabling us to apply Shimada's short vectors algorithm described in section 1.4 in order to determine the set of elements

$$y \in (w_S)^\perp \subset S^\vee$$

satisfying

$$\langle y, y \rangle_{S^\vee} + 2\langle y, c \rangle_{S^\vee} + \langle c, c \rangle_{S^\vee} \leq -2 - \beta.$$

We have seen that this algorithm requires a positive quadratic triple as input data. This triple consists of a Gram matrix of $(w_S)^\perp$, of a column vector, and of a constant. We now explain how to determine such a triple. In order to compute a Gram matrix of $(w_S)^\perp$, we first need to compute a basis of this subspace. An

element $x \in S^\vee$ belongs to $(w_S)^\perp$ if and only if it satisfies

$$\langle x, w_S \rangle_{S^\vee} = 0.$$

Solving this equation for $x = [x_1, \ldots, x_\rho] \in S^\vee$ amounts to determining integers $x_1, \ldots, x_\rho$ such that

$$\begin{bmatrix} x_1 & \ldots & x_\rho \end{bmatrix} G_{S^\vee} \begin{bmatrix} w_1^\vee \\ \vdots \\ w_\rho^\vee \end{bmatrix} = 0. \tag{1.15}$$

and can be done by proceeding as explained at the beginning of the explanations for the implementation of **(a)** in order to obtain a basis for $(w_S)^\perp$. Note that you can also directly use the computer and functions from the SageMath library (or Magma) to do so. Using this basis, we compute a Gram Matrix of $(w_S)^\perp$. That is, we compute the matrix

$$\left[\langle \xi_i, \xi_j \rangle_{S^\vee}\right]_{1 \leq i,j \leq \rho-1}.$$

Denote by $p_\alpha \in S^\vee$ a solution of the equation

$$\langle x, w_S \rangle_{S^\vee} = 1 - \alpha.$$

Such a solution can be obtained using the guidelines we provided in the paragraph dedicated to the implementation of (a). We are now ready to determine to an element $y \in (w_S)^\perp \subset S^\vee$ satisfying

$$\langle y + p_\alpha, y + p_\alpha \rangle_{S^\vee} = -2 - \beta, \tag{1.16}$$

In order to stay in line with the input data format of Shimada's short lattice vectors algorithm, we start by replacing the $=$ sign in

$$\langle y + p_\alpha, y + p_\alpha \rangle_{S^\vee} = -2 - \beta \tag{1.17}$$

by an $\geq$ sign. There is no loss of generality in doing so since the data of the vectors y satisfying the equality will be contained in the set returned by the algorithm. We moreover have to remember that the Gram matrix of $(w_S)^\perp$ is negative define. This fact forces us to multiply both sides of (1.17) by -1 before applying Shimada's short vectors algorithm **ShiVectors**, thus finally bringing us into line with the input data format required by this algorithm. Thus, expanding, arranging, and turning the expression (1.17) into an inequality, we obtain:

$$\langle y, y\rangle_{S^\vee} + 2\langle y, p_\alpha\rangle_{S^\vee} + \langle p_\alpha, p_\alpha\rangle_{S^\vee} + 2 + \beta \geq 0. \tag{1.18}$$

Since y is here assumed to be an element of $(w_S)^\perp$, it can be expressed it as

$$y = y_1\xi_1 + \cdots + y_{\rho-1}\xi_{\rho-1}$$

where the ξ_i are elements of the basis for $(w_S)^\perp$ which has been explicitly computed earlier. The term $2\langle y, p_\alpha\rangle_{S^\vee}$ in (1.18) can then be expressed as:

$$\begin{aligned}
2\langle y, p_\alpha\rangle_{S^\vee} &= 2\langle y_1\xi_1 + \cdots + y_{\rho-1}\xi_{\rho-1}, p_\alpha\rangle_{S^\vee} \\
&= 2(y_1\langle \xi_1, p_\alpha\rangle_{S^\vee} + \cdots + y_{\rho-1}\langle \xi_{\rho-1}, p_\alpha\rangle_{S^\vee}) \\
&= 2\begin{bmatrix} y_1 & \dots & y_{\rho-1}\end{bmatrix}\begin{bmatrix} \langle \xi_1, p_\alpha\rangle_{S^\vee} \\ \vdots \\ \langle \xi_{\rho-1}, p_\alpha\rangle_{S^\vee}\end{bmatrix} \\
&= 2yP
\end{aligned}$$

where P is the $(\rho-1)$-sized column vector thus defined as

$$P = \begin{bmatrix} \langle \xi_1, p_\alpha\rangle_{S^\vee} \\ \vdots \\ \langle \xi_{\rho-1}, p_\alpha\rangle_{S^\vee}\end{bmatrix}.$$

Denoting by G_w the Gram matrix of $(w_S)^\perp$, we see that we established that the

inequality

$$\langle y + p_\alpha, y + p_\alpha \rangle_{S^\vee} \geq -2 - \beta$$

is equivalent to

$$y\, G_w\, y^t + 2yP + c \geq 0$$

where

$$y = [y_1, \ldots, y_{\rho-1}] \qquad \text{and} \qquad c = \langle p_\alpha, p_\alpha \rangle_{S^\vee} + 2 + \beta.$$

By the Hodge Index Theorem, the Gram matrix G_w of $(w_S)^\perp$ is negative definite. We thus replace it by its negative $-G_w$ and do the same for P and c. We hence obtain an inequality involving a positive quadratic form on the left-hand side, forming an expression fully in line with the input data format required by Shimada's short vectors algorithm:

$$y\, (-G_{w_S^\perp})\, y^t + 2y(-P) + (-c) \leq 0$$

The positive quadratic triple to be used as input data into Shimada's short lattice vectors enumerator **ShiVectors** is therefore given by:

$$\left[-G_{w_S^\perp}, -L, -c\right] = \left[-G_{w_S^\perp}, -\begin{bmatrix} \langle \xi_1, p_\alpha \rangle_{S^\vee} \\ \vdots \\ \langle \xi_{\rho-1}, p_\alpha \rangle_{S^\vee} \end{bmatrix}, -\langle p_\alpha, p_\alpha \rangle_{S^\vee} - 2 - \beta\right].$$

Executing this algorithm produces the set of all of elements $q \in (w_S)^\perp$ such that

$$\langle q + p_\alpha, q + p_\alpha \rangle_{S^\vee} \geq -2 - \beta,$$

from which can be extracted the set of elements $q \in (w_S)^\perp$ satisfying

$$\langle q + p_\alpha, q + p_\alpha \rangle_{S^\vee} = -2 - \beta.$$

Fix such an element, say q_0, and let

$$v_S = q_0 + p_\alpha$$

The element v_S clearly satisfies

$$\langle v_S, v_S \rangle_{S^\vee} = -2 - \beta.$$

Since $q_0 \in (w_S)^\perp$, we moreover have

$$\langle q_0, w_S \rangle_{S^\vee} = 0.$$

Since p_α is furthermore assumed to belong to the solution set of

$$\langle x, w_S \rangle_{S^\vee} = 1 - \alpha,$$

we have

$$\begin{aligned} \langle v_S, w_S \rangle_{S^\vee} &= \langle q_0 + p_\alpha, w_S \rangle_{S^\vee} \\ &= 0 + \langle p_\alpha, w_S \rangle_{S^\vee} = 1 - \alpha. \end{aligned}$$

Recall that an element $v_R \in R^\vee$ such that

$$\alpha = \langle v_R, w_R \rangle_{R^\vee} \qquad \text{and} \qquad \beta = \langle v_R, v_R \rangle_{R^\vee}$$

is assumed to be given since the beginning of **Step n°4**.

Step n°5: Denote by $v_S^{\mathbb{L}\otimes\mathbb{Q}}$ (resp. $v_R^{\mathbb{L}\otimes\mathbb{Q}}$) the image of v_S (resp. v_R) under the transformation which expresses an element of $S^\vee \subset \mathbb{L}$ (resp $R^\vee \subset \mathbb{L}$) in terms of the standard basis of $\mathbb{L} \otimes \mathbb{Q}$. Assume that

$$v_S^{\mathbb{L}\otimes\mathbb{Q}} + v_R^{\mathbb{L}\otimes\mathbb{Q}} \in \mathbb{L}.$$

That is, assume that $v_S^{\mathbb{L}\otimes\mathbb{Q}} + v_R^{\mathbb{L}\otimes\mathbb{Q}}$ has integer coordinates. Note that the Weyl

vector w can be expressed as

$$w = w_S^{\mathbb{L}\otimes\mathbb{Q}} + w_R^{\mathbb{L}\otimes\mathbb{Q}}.$$

We then have

$$\begin{aligned}
\left\langle v_S^{\mathbb{L}\otimes\mathbb{Q}} + v_R^{\mathbb{L}\otimes\mathbb{Q}}, w \right\rangle_{\mathbb{L}} &= \left\langle v_S^{\mathbb{L}\otimes\mathbb{Q}} + v_R^{\mathbb{L}\otimes\mathbb{Q}}, w_S^{\mathbb{L}\otimes\mathbb{Q}} + w_R^{\mathbb{L}\otimes\mathbb{Q}} \right\rangle_{\mathbb{L}} \\
&= \left\langle v_S^{\mathbb{L}\otimes\mathbb{Q}}, w_S^{\mathbb{L}\otimes\mathbb{Q}} \right\rangle_{\mathbb{L}} + \left\langle v_R^{\mathbb{L}\otimes\mathbb{Q}}, w_R^{\mathbb{L}\otimes\mathbb{Q}} \right\rangle_{\mathbb{L}} \\
&= \langle v_S, w_S \rangle_{S^\vee} + \langle v_R, w_R \rangle_{R^\vee} \\
&= 1 - \alpha + \alpha \\
&= 1
\end{aligned}$$

and

$$\begin{aligned}
\left\langle v_S^{\mathbb{L}\otimes\mathbb{Q}} + v_R^{\mathbb{L}\otimes\mathbb{Q}}, v_S^{\mathbb{L}\otimes\mathbb{Q}} + v_R^{\mathbb{L}\otimes\mathbb{Q}} \right\rangle_{\mathbb{L}} &= \left\langle v_S^{\mathbb{L}\otimes\mathbb{Q}}, v_S^{\mathbb{L}\otimes\mathbb{Q}} \right\rangle_{\mathbb{L}} + \left\langle v_R^{\mathbb{L}\otimes\mathbb{Q}}, v_R^{\mathbb{L}\otimes\mathbb{Q}} \right\rangle_{\mathbb{L}} \\
&= \langle v_S, v_S \rangle_{S^\vee} + \langle v_R, v_R \rangle_{S^\vee} \\
&= -2 - \beta + \beta \\
&= -2.
\end{aligned}$$

Since $\langle v_R, v_R \rangle_{R^\vee}$ is assumed to belong to n_R, and since the elements of this set satify by definition of n_R the inequalities

$$-2 < c \leq 0,$$

one can readily deduce from the equality

$$\langle v_S, w_S \rangle_{S^\vee} + \langle v_R, w_R \rangle_{R^\vee} = 1$$

established above that

$$\langle v_S, v_S \rangle_{S^\vee} < 0$$

holds. Consequently,

$$v_S^{\mathbb{L}\otimes\mathbb{Q}} + v_R^{\mathbb{L}\otimes\mathbb{Q}} \in \Delta_w,$$

as desired, where we recall that

$$\Delta_w = \{x \in \mathbb{L} \mid \langle x, x\rangle_{\mathbb{L}} = -2,\ \langle x, w\rangle_{\mathbb{L}} = 1,\ \langle x_S, x_S\rangle_{S^\vee} < 0\}\,.$$

1.5.2 Procedure SetOfWalls

We have seen in the previous section how to compute the set Δ_w from the input data of a Weyl vector $w \in \mathbb{L}$ of a $\mathcal{P}_S$-chamber D. Moreover, proposition 12 from section 1.2 states that $\mathrm{pr}_S(\Delta_w)$ is a defining set of D. Shimada's algorithm 3.17 from [19] enables us to compute the primitively minimal defining set of D, that is, the set $\Omega(D)$ of walls of D, from the input data of $\mathrm{pr}_S(\Delta_w)$.

We follow the structure outlined by Shimada in his article and provide additional details to enable our readers to implement this algorithm without hassle. Our implementation **SetOfWalls** of this algorithm is available for download on our website. We briefly go back within the framework of an unspecified even hyperbolic lattice L with a fixed positive cone $\mathcal{P}_L$. Let D be a $\mathcal{P}_L$-chamber. Recall that a hyperplane $(v)^\perp$ of $\mathcal{P}_L$ is called of wall of D if

$$(v)^\perp \cap \mathrm{Int}(D) = \emptyset$$

holds and if $(v)^\perp \cap D$ contains a non-empty open subset of $(v)^\perp$. We begin with the following lemma due to Shimada.

Lemma 18. *Let L be an even hyperbolic lattice. Assume that a defining set Δ of a chamber D has the property that any two of its distinct elements $v_1 \neq v_2$ satisfy $(v_1)^\perp \neq (v_2)^\perp$. Then the following statements hold for any element $v \in \Delta$,*

(i) If $\Delta \setminus \{v\}$ does not span $L \otimes \mathbb{R}$, then $(v)^\perp$ is a wall of D and

(ii) the hyperplane $(v)^\perp$ is a wall of D if and only if $\Sigma_L(\Delta) \neq \Sigma_L(\Delta \setminus \{v\})$.

This lemma provides criteria to determine whether an element of a defining set of a chamber D has an orthogonal complement defining a wall of D. Let Δ be a defining set of a $\mathcal{P}_S$-chamber D. The assumption that any two of the distinct elements $v_1 \neq v_2$ of Δ satisfy $(v_1)^\perp \neq (v_2)^\perp$ at the beginning of the lemma takes its roots in the definition 2 of a defining set. Indeed, this definition does not prevent the occurrence of distinct elements having the same orthogonal complement, thus potentially defining the same wall. Such a redundancy is pointless and should be avoided. In practice, situations in which this issue arises are always caused by of elements $v, v' \in \Delta$ related by an equality of the form

$$v = kv' \tag{1.19}$$

where $k \in \mathbb{Z}$. The best course of action to prevent their occurrence consists in dividing the coefficients of each element of Δ by their greatest common divisor. Indeed, elements v, v' related by an equality such as (1.19) satisfy

$$\frac{v}{\gcd(v)} = \pm\frac{v'}{\gcd(v')}$$

where we denote by $\gcd(v)$ the greatest common divisor of the coordinates of an element $v \in S^\vee$. We thus substitute the set

$$\Delta' = \{v/\gcd(v) \mid v \in \Delta\},$$

to Δ and make sure that if $v \in \Delta'$ then $-v \notin \Delta'$. We proceed to points (i) and (ii) of the lemma. Enforcing point *(i)* of Lemma 18 is straightforward: Given an element $v \in \Delta'$, we can use SageMath lattice features to determine whether the sublattice of $S^\vee$ spanned by $\Delta' \setminus \{v\}$ has rank equal to $\mathrm{rank}(S)$. We explain how to do this on our website. If this is the case, then the lemma states that $(v)^\perp$ is not a wall of D. Otherwise, the lemma tells us that $(v)^\perp$ is a wall of D. Let us take a closer look to (ii), which states that given an element $v \in \Delta$, the

hyperplane $(v)^{\perp}$ is a wall of $\mathcal{P}_L$-chamber D if and only if

$$\Sigma_S(\Delta) \neq \Sigma_S(\Delta \setminus \{v\}).$$

First, we recall that the positive cone $\Sigma_S(\Delta)$ associated with Δ is defined as

$$\Sigma_S(\Delta) = \{x \in S \otimes \mathbb{R} \mid \forall v \in \Delta,\, \langle x, v\rangle_S \geq 0\}$$

and recall that we have by definition $D = \Sigma_S(\Delta) \cap \mathcal{P}_S$. To understand the statement of point (ii), let $p \in \Delta$ be such that $(p)^{\perp}$ is not a wall of D. Since $(p)^{\perp}$ is not a wall of D, the data of p is irrelevant and unecessary to define the chamber D. Hence, we have

$$D = \Sigma_S(\Delta \setminus \{p\}) \cap \mathcal{P}_S.$$

and the positive cone $\Sigma_S(\Delta \setminus \{p\})$ cone associated with $\Delta \setminus \{p\}$ coincides with the positive cone $\Sigma_S(\Delta)$ associated with Δ. Let us turn things over and assume that $p \in \Delta$ is such that $(p)^{\perp}$ is a wall of the chamber D. Then, there exist at least an element $v_0 \in S \otimes \mathbb{R}$ such that

$$\langle v_0, q\rangle_S \geq 0$$

for all $q \in \Delta \setminus \{p\}$ but satisfying

$$\langle v_0, p\rangle_S < 0.$$

Thus,

$$v_0 \in \Sigma_S(\Delta \setminus \{p\})$$

and there is a strict inclusion

$$\Sigma_S(\Delta) \subset \Sigma_S(\Delta \setminus \{p\}).$$

This observation also reveals the two following important facts:

- If $(p)^\perp$ is not a wall of D, then the solution x_{sol} obtained by minimizing the function $f_p(x) = \langle x, p\rangle_S$ subject to the constraints $\langle x, q\rangle_S \geq 0$ for all $q \in \Delta \setminus \{p\}$ satisfies $f_p(x_{\text{sol}}) \geq 0$.

- If $(p)^\perp$ is a wall of D, the solution x_{sol} obtained by minimizing the function $f_p(x) = \langle x, p\rangle_S$ subject to the constraints $\langle x, q\rangle_S \geq 0$ for all $q \in \Delta \setminus \{p\}$ must satisfy $f_p(x_{\text{sol}}) = d$ with d negative and possibly unbounded toward infinity.

Performing this check can be done using **linprog** from **scipy.optimize**. We explain how we proceeded to do so in an online section. We now have all the tools in hand to introduce our user-friendly version of Shimada's Algorithm 5.11 from [19] which encompasses all the material required to obtain the set of walls of a chamber from the only input of its Weyl vector. Procedure **SetOfWalls:** Let D be a $\mathcal{P}_S$-chamber with Weyl vector w.

Step n°1 - Using the procedure **DeltaW**, compute the set Δ_w.

Step n°2 - Compute the set $\Delta' = \{v/\gcd(v) \mid v \in \Delta\}$.

Step n°3 - For each $p \in \Delta'$, proceed as follows: Determine whether the sublattice of $S^\vee$ spanned by $\Delta' \setminus \{p\}$ has rank equal to $\text{rank}(S^\vee)$, where the latter is the Picard number of S. If this is the case, then $(p)^\perp$ is not a wall of D by lemma 18. Delete p from Δ'. Otherwise, the lemma tells us that $(p)^\perp$ is a wall of D. Then, use **linprog** from **scipy.optimize** to solve the following optimization problem: Minimize the function

$$f_p(x) = \langle x, p\rangle_S$$

subject to the constraints

$$\langle x, q\rangle_S \geq 0$$

for all $q \in \Delta \setminus \{p\}$ and denote by x_{opt} the resulting solution.

- If $f_p(x_{\text{opt}}) = 0$, then $(p)^{\perp}$ is not a wall of D. Delete p from Δ'.
- If $f_p(x_{\text{opt}})$ is strictly negative and possibly unbounded toward infinity, then $(p)^{\perp}$ is a wall of D.

1.6 Computation of generators of Aut(X) - Background

The article [19] in which Shimada introduced his pioneering approach to Borcherds' method was issued almost a decade ago. Nonetheless, it was not until this thesis that a general application framework of application for Borcherds' method was identified and explicitly stated. This is undoubtedly one of the reasons that, outside of Shimada's implementation which has never been released to the public, no trace of an implementation of any kind of the Borcherds' method could be found on the internet until the arrival of this thesis in 2022. It was to be expected: Without an algorithmically testable framework of application, what would be the point of implementing Borcherds' method? We put an end to this unfortunate situation in this section:

- First, we assemble Shimada's puzzle by putting together the pieces which can be found in his article [19] to exhibit a general framework of application for Borcherds' method.
- Second, from the knowledge of this framework, we determine a concrete criterion to determine whether Borcherds' method can be applied to a given $K3$ surface and produce a generating set of its automorphism group.

We thus start by acting as investigators motivated by the goal of exhibiting a general framework of application for Borcherds' method from the information contained in Shimada's article. Before proceeding further, let us get things straight about the notations involved in this section:

- We denote by X a complex algebraic $K3$ surface.
- We denote by S the Néron-Severi group $\mathrm{NS}(X)$ of X.

- We denote by $\mathcal{P}_S$ the positive cone of X, i.e., the connected component of

$$\{x \in S \mid \langle x, x\rangle_S > 0\}$$

of S containing ample classes.

- We denote by $\mathrm{Aut}(X)$ the automorphism group of X.

- We denote by T the transcendental lattice of X. That is, T is the orthogonal complement of S in

$$H^2(X, \mathbb{Z}) \simeq U^3 \oplus E_8(-1)^2.$$

- We denote by $\mathrm{Nef}(X)$ the numerically effective cone of X. This cone is often referred to as the Nef cone of X. More appropriate, we use the notation $\mathcal{N}_X$ in order to denote the intersection $\mathrm{Nef}(X) \cap \mathcal{P}_S$.

- We denote by $S^\vee/S$ the discriminant group of S and let

$$q_S : S^\vee/S \longmapsto \mathbb{Q}\,/\,2\mathbb{Z}$$

be its associated quadratic form.

- We denote by $T^\vee/T$ the discriminant group of T and

$$q_T : T^\vee/T \longmapsto \mathbb{Q}\,/\,2\mathbb{Z}$$

will denote the associated quadratic form.

- We denote by $O(S)$, $O(T)$, $O(q_S)$ and $O(q_T)$ the respective groups of isometries of the lattices S, T and of the disc. groups $S^\vee/S$, $T^\vee/T$.

- Denote by $O^+(S)$ the subgroup of $O(S)$ preserving $\mathcal{P}_S$.

- The subgroup of $O^+(S)$ preserving $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is denoted by

$$\mathrm{Aut}(\mathrm{Nef}(X) \cap \mathcal{P}_S) = \left\{ g \in O^+(S) \mid \mathcal{N}_X^g = \mathcal{N}_X \right\}.$$

1.6.1 Scope of application of Borcherds' method

We still have to mention the two following results that will be useful to us:

- It is well-known that an isometry of S (resp. T) induces an isometry of $S^\vee/S$ (resp $T^\vee/T$) in a canonical way, so that there are natural homomorphisms

$$\eta_S : O(S) \longrightarrow O(S^\vee/S) \quad \text{and} \quad \eta_T : O(T) \longrightarrow O(T^\vee/T).$$

- As indicated at the beginning of Shimada's [19, section 5], there exists an isomorphism

$$\delta : (S^\vee/S, q_S) \longrightarrow (T^\vee/T, -q_T)$$

of discriminant forms which in turns induces an isomorphism

$$\psi : O(S^\vee/S) \longrightarrow O(T^\vee/T)$$

of the groups of isometries of $S^\vee/\,S$ and of $T^\vee/\,T$.

The situation can be summarized as follows

$$\begin{array}{ccccc} O(T) & \longleftarrow & \mathrm{Aut}(X) & \xrightarrow{\varphi_X} & O(S) \\ \Big\downarrow{\scriptstyle \eta_T} & & & & \Big\downarrow{\scriptstyle \eta_S} \\ O(T^\vee/\,T) & & \xleftarrow{\quad\quad\psi\quad\quad} & & O(S^\vee/\,S) \end{array}$$

We start by recalling a well-known piece of theoretical material in the field of study of $K3$ surfaces: The famous Torelli theorem states that to each effective Hodge isometry

$$\Phi : H^2(X, \mathbb{Z}) \to H^2(X, \mathbb{Z})$$

can be uniquely associated an automorphism

$$f : X \to X$$

such that

$$\Phi = f^*.$$

Let $\omega \in T \otimes \mathbb{C}$ be a non-zero holomorphic 2-form and define

$$C_T = \left\{ g \in O(T) \mid \exists \lambda \in \mathbb{C}^\times \text{ such that. } \omega^g = \lambda \omega \right\}.$$

By definition of C_T and of the morphisms η_T and η_S introduced earlier, an element $g \in O^+(S)$ extends to an effective Hodge isometry if and only if

$$\psi(\eta_S(g)) \in \eta_T(C_T).$$

The following result due to Piatetski-Shapiro & Shafarevich [14] and stated in [19, Theorem 7.1] will be central for the continuation of our study:

Proposition 19. *Via the natural actions of Aut(X) on the lattices S and T, the automorphism group Aut(X) is identified with*

$$\{(g_S, g_T) \in \mathit{Aut}(\mathit{Nef}(X) \cap \mathcal{P}_S) \times C_T \mid \psi(\eta_S(g_S)) = \eta_T(g_T)\}\,.$$

Since $O(q_T)$ is finite, the subgroup

$$\mathbf{H} := \left\{g_S \in O^+(S) \mid \psi(\eta_S(g_S)) \in \eta_T(C_T)\right\}$$

of $O^+(S)$ has finite index.

It should be understood from the first part of this theorem that a pair (g_S, g_T) can be associated with each $g \in \mathrm{Aut}(X)$ and that its elements g_S, g_T satisfy

- $g_S \in \mathrm{Aut}(\mathrm{Nef}(X) \cap \mathcal{P}_S) \subset O(S)$,
- $g_T \in C_T \subset O(T)$,
- $\psi(\eta_S(g_S)) = \eta_T(g_T)$.

That is, the image of the morphism

$$\varphi_X : \mathrm{Aut}(X) \longrightarrow O(S)$$

satisfies

$$\mathrm{Im}(\varphi_X) \subset \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$$

where

$$\begin{aligned}\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S) &= \{g \in \mathbf{H} \mid g \ \text{ preserves } \ \mathrm{Nef}(X) \cap \mathcal{P}_S\} \\ &\subset \mathrm{Aut}(\mathrm{Nef}(X) \cap \mathcal{P}_S).\end{aligned}$$

For the remainder of this section, we ask the reader to keep in mind the fact that, in the framework of a complex algebraic $K3$ surface X, Borcherds' method is a procedure which produces a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$. Following proposition 19, Shimada introduced in [19] the following corollary in [19,

Corollary 7.2] which formalizes the consequences of proposition 19 and brings an additional characterization of $\mathrm{Ker}(\varphi_X)$ to the table:

Corollary 20. *The kernel of φ_X is isomorphic to $\mathrm{Ker}(\eta_T) \cap C_T$. The image of φ_X is isomorphic to*

$$\begin{aligned} \mathit{Aut}_{\mathbf{H}}(\mathit{Nef}(X) \cap \mathcal{P}_S) &= \{g \in \mathbf{H} \mid \mathcal{N}_X^g = \mathcal{N}_X\} \\ &\subset \mathit{Aut}(\mathit{Nef}(X) \cap \mathcal{P}_S) \end{aligned}$$

Shimada also introduced the following proposition in section 8.1 of [19]:

Proposition 21. *If $\rho_X < 20$ and the period ω_X of X is very general in $T \otimes \mathbb{C}$, then*

$$C_T = \{\pm 1\}.$$

Combining this result to the characterization of $\mathrm{Ker}(\varphi_X)$ provided in corollary 20 enables us to assert that

$$\mathrm{Ker}(\varphi_X) \subset \{\pm 1\}$$

holds whenever the $K3$ surface X under study is very general and has a Picard number ρ_X satisfying

$$\rho_X < 20.$$

Assume that $-1 \notin \mathrm{Ker}(\eta_T)$ also holds, so that $\mathrm{Ker}(\varphi_X) = \{1\}$. In this case, the morphism φ_X is injective. Under this assumption, it is clear that the image of the morphism

$$\varphi_X : \mathrm{Aut}(X) \longrightarrow O(S)$$

then satisfies

$$\mathrm{Im}(\varphi_X) \simeq \mathrm{Aut}(X).$$

Keeping in mind that corollary 20 states that

$$\mathrm{Im}(\varphi_X) \simeq \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$$

we hence obtain by transitivity that

$$\mathrm{Aut}(X) \simeq \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S).$$

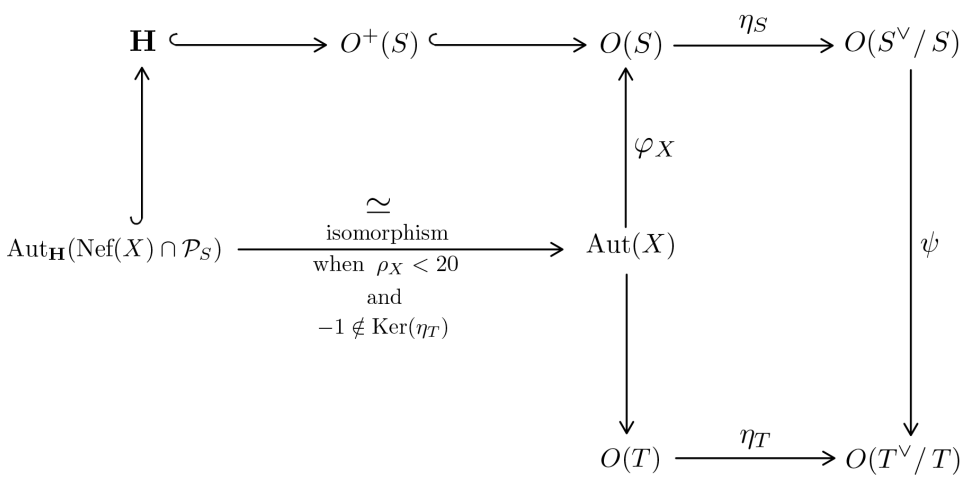
The pieces of the puzzle can then all be put together:

Theorem 22. *If X is very general (we will always assume that it is the case), satisfies $\rho_X < 20$ and $-1 \notin \mathrm{Ker}(\eta_T)$, then there is an isomorphism*

$$\mathit{Aut}(X) \simeq \mathit{Aut}_{\mathbf{H}}(\mathit{Nef}(X) \cap \mathcal{P}_S).$$

The above theorem enables us to exhibit a general framework of application of the method for the computation of automorphism groups: Borcherds' method returns a generating set of $\mathrm{Aut}(X)$ whenever X is a complex $K3$ surface of Picard number $\rho_X < 20$ satisfying $-1 \notin \mathrm{Ker}(\eta_T)$.

The following figure provides a clear view of the situation:



Keep in mind that Borcherds' method, by design, produces a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$. This is why a generating set of $\mathrm{Aut}(X)$ can be obtained for complex $K3$ surfaces satisfying the above-mentioned conditions. We will soon provide in this section a criterion to determine whether the condition $-1 \notin \mathrm{Ker}(\eta_T)$ holds. Note that also our program **KerChecker** is available on our website and will automatically perform this check. Borcherds' method to compute generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$, as presented by Shimada ten years ago, is therefore not limited to a handful of special cases of $K3$ surfaces X for which it will provide generators of $\mathrm{Aut}(X)$. There is a clear general framework of application for complex $K3$ surfaces, opening up very broad prospects for study. Although this framework was not explicitly apparent in Shimada's article [19], all the material used above could be found there. We still have to tackle two issues in order to be able to take advantage of the theorem 22:

- **Issue n°1:** We need to provide Borcherds' method with a generalized membership criterion for $\mathbf{H}$.

- **Issue n°2:** We need to provide a concrete criterion to check whether

$$-1 \notin \mathrm{Ker}(\eta_T)$$

  holds. Click here for practical details regarding this matter, this webpage contains an online version of the content of the section 1.6.3.

1.6.2 Finding a generalized membership criterion

We start by providing a solution to the **Issue n°1:** Let X be a $K3$ surface X satisfying the conditions of theorem 22. Let $g \in \mathrm{Aut}(X)$ and consider the associated pair (g_S, g_T) provided by proposition 19. The latter also states that the element

$$g_S \in \mathrm{Aut}(\mathrm{Nef}(X) \cap \mathcal{P}_S) \subset O^+(S)$$

satisfies

$$\psi(\eta_S(g_S)) \in \eta_T(C_T),$$

that is, $g_S \in \mathbf{H}$. By proposition 21, we have

$$C_T = \{\pm 1\}.$$

The group $\mathbf{H}$ can then be expressed as

$$\mathbf{H} = \left\{h_S \in O^+(S) \mid \psi(\eta_S(h_S)) \in \{\pm 1\}\right\}.$$

Since

$$\psi : O(q_S) \longrightarrow O(q_T)$$

is an isomorphism, the definition of $\mathbf{H}$ can be further refined as

$$\mathbf{H} = \left\{h_S \in O^+(S) \mid \eta_S(h_S) \in \{\pm 1\}\right\},$$

where we recall that

$$\eta_S : O(S) \longrightarrow O(q_S)$$

is the natural morphism which turns isometries of S into isometries of its discriminant group $S^\vee/S$. Thus, an element $g_S \in \mathrm{Aut}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ such that

$$\eta_S(g_S) \in \{\pm 1\}$$

can be associated with each automorphism $g \in \mathrm{Aut}(X)$. Conversely, if we let $q \in \mathrm{Aut}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ be such that $\eta_S(q) \in \{\pm 1\}$, then the correspondence provided by proposition 19 enables us to exhibit an element $h \in \mathrm{Aut}(X)$ such that $q = h_S$, where

$$(h_S, h_T)$$

is the pair associated with h by this correspondence. A precise characterization of the elements of $\mathrm{Aut}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ originating from automorphisms thus becomes apparent, and can be formalized in the following proposition:

Proposition 23. *Assume that $\rho_X < 20$ and that $-1 \notin \mathrm{Ker}(\eta_T)$. Then an element $h \in Aut(Nef(X) \cap \mathcal{P}_S)$ emanates from an automorphism $g \in Aut(X)$, i.e., satisfies $h = g_S$ by the identification of proposition 19 if and only if*

$$\eta_S(h) \in \{\pm 1\}\,.$$

That is, $h \in \mathbf{H}$ if and only if its acts on the discriminant group $S^\vee/S$ as $\pm Id$.

Note that a $(\rho \times \rho)$-sized invertible matrix of the form

$$\begin{pmatrix} a_{11} & \dots & a_{1\rho} \\ \vdots & \ddots & \vdots \\ a_{\rho 1} & \dots & a_{\rho\rho} \end{pmatrix} \qquad a_{ij} \in \mathbb{Z}, \quad 1 \leq i, j \leq \rho.$$

can be associated with each element of $O(S)$, in a framework of a given basis $\mathcal{B}$. Such matrices, say $g \in \mathrm{GL}_\rho(\mathbb{Z})$, act from the right on ρ-sized row vectors representing elements of S, e.g.,

$$v \longmapsto vg,$$

where $v \in S$. Such matrices satisfy by definition

$$gG_Sg^T = G_S$$

where we recall that G_S denote the Gram matrix of S with respect to $\mathcal{B}$ and where g^T denotes the transpose of the matrix g. Note that our previous discussion enables us to assert that whenever the conditions of theorem 22 hold, the subgroup $\mathbf{H}$ of $O^+(S)$ can be expressed as

$$\mathbf{H} = \left\{h_S \in O^+(S) \mid \eta_S(h_S) \in \{\pm 1\}\right\}.$$

In order to obtain a membership criterion for $\mathbf{H}$, we thus have to be able to:

- Determine whether an element $g \in \mathrm{GL}_\rho(\mathbb{Z})$ belongs to $O^+(S)$.
- Determine whether an element of $O^+(S)$ acts as $\pm\mathrm{Id}$ on $S^\vee/S$.

Dealing with the first point is an easy task: Let $g \in \mathrm{GL}_\rho(\mathbb{Z})$. Then $g \in O^+(S)$ if and only if $g \in O(S)$ and if g preserves $\mathcal{P}_S$. That is, g must satisfy

$$gG_Sg^T = G_S$$

and determining whether g preserves $\mathcal{P}_S$ can be done by taking any ample class $a_0 \in \mathcal{P}_S$ and checking whether

$$(a_0g)G_Sa_0^T > 0,$$

i.e., whether the image of an ample class $a_0 \in \mathcal{P}_S$ by g is still contained in $\mathcal{P}_S$. Note that an element $g \in O^+(S)$ acts as $\pm\mathrm{Id}$ on the discriminant group $S^\vee/S$ of S if and only if there exists $\epsilon \in \{\pm 1\}$ such that

$$g^*t = \epsilon\, t$$

holds for all generators t of $S^\vee/S$, where g^* denotes the transformation of $S^\vee/S$ naturally associated with

$$g \in O^+(S) \subset O(S)$$

by the natural morphism which turns elements of $O^+(S)$ into transformations of $O(S^\vee/S)$. It is well-known that the columns $b_i = \mathrm{col}_i(G_S^{-1})$ of the inverse of the matrix G_S can be taken as representatives of the generators of $S^\vee/S$. Thus, an element $g \in O^+(S)$ acting as $+\mathrm{Id}$ or $-\mathrm{Id}$ on $S^\vee/S$ must either satisfy the conditions

$$b_ig - b_i \in \mathbb{Z}^{\rho_X} \qquad \text{for all} \qquad 1 \leq i \leq \rho_X$$

or the conditions

$$b_i g - b_i \in \mathbb{Z}^{\rho_X} \qquad \text{for all} \qquad 1 \leq i \leq \rho_X$$

This conditions can be reformulated as: An element $g \in O^+(S)$ acting as $+\mathrm{Id}$ or $-\mathrm{Id}$ on $S^\vee / S$ must satisfy either

$$G_S^{-1} g - G_S^{-1} \in M_\rho(\mathbb{Z}) \qquad \text{or} \qquad G_S^{-1} g + G_S^{-1} \in M_\rho(\mathbb{Z})$$

where $M_\rho(\mathbb{Z})$ denote the group of $(\rho \times \rho)$-sized matrices with integer coefficients. We thus established the following proposition:

Proposition 24. *Assume that the conditions of theorem 22 are satisfied. An element $g \in O(S)$ belongs to $\mathbf{H}$ if and only if*

- $g\, G_S\, g^T = G_S$
- $a_0\, g\, G_S\, a_0^T > 0$ *for an ample class $a_0 \in NS(X)$*
- *Either (a) or (b) below hold:*

  (a) $G_S^{-1} g - G_S^{-1} \in M_\rho(\mathbb{Z})$

  (b) $G_S^{-1} g + G_S^{-1} \in M_\rho(\mathbb{Z})$

Our procedure **MemberCrit** is a direct implementation of this proposition: It takes as input an invertible matrix with integer coefficients and outputs a Boolean value True or False depending on whether the matrix used as input data belongs to $\mathbf{H}$ or not.

1.6.3 Checking the kernel condition

We start by recalling that the transcendental lattice T associated with X is the orthogonal complement of $S = \mathrm{NS}(X)$ in the rank 22 lattice

$$H^2(X, \mathbb{Z}) \simeq U^3 \oplus E_8(-1)^2.$$

We also recall that we denote by η_T is the natural morphism

$$\eta_T : O(T) \longrightarrow O(T^\vee / T).$$

which turns isometries of T into isometries of its discriminant group. We note that the rank of T is equal to $22 - \rho$, where $\rho = \mathrm{rank}(S)$. If we assume a basis fixed for T, then an element of $\mathrm{GL}_{22-\rho}(\mathbb{Z})$ can be associated with each transformation of $O(T)$. The element $-1 \in O(T)$ can thus be viewed as the matrix $-\mathrm{Id}_{22-\rho}$. The latter will be denoted by $-\mathrm{Id}$ for the remainder of this section. In order to find a way to check whether $-1 \notin \mathrm{Ker}(\eta_T)$, we are going to use the same trick that we used to derive a membership criterion for $\mathbf{H}$. Assume that $-\mathrm{Id} \in \mathrm{Ker}(\eta_T)$, i.e., that the matrix $-\mathrm{Id} \in O(T)$ acts as the identity element of $O(T^\vee / T)$ via the natural morphism η_T. Then $-\mathrm{Id}$ must preserve each generator of the discriminant group $T^\vee / T$, where $1 \leq i \leq 22 - \rho$. Keeping in mind that representatives of basis elements of $T^\vee / T$ are obtained by taking columns of G_T^{-1}, the inverse of the Gram matrix G_T of T, this conditions amounts to

$$2G_T^{-1} \in M_{22-\rho}(\mathbb{Z}).$$

Thus, if

$$2G_T^{-1} \notin M_{22-\rho}(\mathbb{Z}),$$

then

$$-\mathrm{Id} \notin \mathrm{Ker}(\eta_T).$$

Proposition 25. *Let T be the transcendental lattice of X, that is, T is the orthogonal complement of $S := \mathrm{NS}(X)$ in $H^2(X, \mathbb{Z}) \simeq U^3 \oplus E_8(-1)^2$. Consider the natural morphism $\eta_T : O(T) \longrightarrow O(T^\vee / T)$ and let G_T be the Gram matrix of T. The following statement holds:*

$$2G_T^{-1} \notin M_{22-\rho}(\mathbb{Z}) \Longrightarrow -Id \notin \mathrm{Ker}(\eta_T)$$

where $\rho = \mathrm{rank}(S)$.

Assuming that X has Picard number $\rho_X \leq 17$, our procedure **KerChecker** uses the input data of an embedding of S into either

$$U \oplus E_8(-1) \qquad \text{or into} \qquad U \oplus E_8(-1) \oplus E_8(-1),$$

computes a Gram matrix G_T of T with respect to a fixed basis, and then performs the above-mentioned check. **KerChecker** outputs **True** whenever

$$-\mathrm{Id} \notin \mathrm{Ker}(\eta_T)$$

holds, and **False** when

$$-\mathrm{Id} \in \mathrm{Ker}(\eta_T).$$

Click here for more details on the practical and computer-based side of things regarding the procedure **KerChecker** and more generally, regarding the scope of application of Borcherds' method.

```
sage: load('init_emb_no_remarks.sage')

      ✓ The embedding of NS(X) into U+E8(-1) used as input data is primitive !

      ✓ The ECL heap-size limit has been removed !

      ✓ A suitable global namespace has successfully been defined !

      ✓ Up to 20 workers will be mobilized by Pool during operations involving process-based parallelism !

sage: GramMatS
[ 0  1  0]
[ 1 -2  0]
[ 0  0 -2]
sage: load('ker_checker.sage')

*****************************************************************************************************

                              ⚠  ⚠  ⚠  ⚠  ⚠   ⚠  ⚠  ⚠  ⚠  ⚠
                              ⚠  ⚠  ⚠  ⚠  WARNING  ⚠  ⚠  ⚠  ⚠
                              ⚠  ⚠  ⚠  ⚠  ⚠   ⚠  ⚠  ⚠  ⚠  ⚠

      ⚠ -Id BELONGS to Ker(η) !

      ⚠  It cannot be guaranteed that the Borcherds' method will return a generating set of Aut(X) !

*****************************************************************************************************
sage: BOOL_KERCHECKER
False
sage:
```

1.7 Borcherds' method

Please note that an entire section of K3surfaces.com is devoted to the practical and computer-based side of things regarding Borcherds' method. Click here for more details regarding this matter.

Let X be a $K3$ surface over the complex numbers. Assume that X has Picard number ρ_X and fix a primitive embedding

$$\iota : S \hookrightarrow \mathbb{L}$$

of $S = \mathrm{NS}(X)$ into an even hyperbolic lattice $\mathbb{L}$ chosen as recommended in the following table:

Picard number ρ_X	Recommended ambient lattice
$1 \leq \rho_X < 10$	$U \oplus E_8(-1)$
$10 \leq \rho_X < 18$	$U \oplus E_8(-1) \oplus E_8(-1)$
$18 \leq \rho_X < 20$	$U \oplus E_8(-1) \oplus E_8(-1) \oplus E_8(-1)$

We moreover assume that the pritimive embedding $\iota : S \hookrightarrow \mathbb{L}$ is such that

$$\iota(\mathcal{P}_S) \subset \mathcal{P}_{\mathbb{L}}.$$

Using the material discussed in the previous sections, we proceed as follows:

- Following the steps explained in section 1.1.2, we set a $\mathcal{P}_{\mathbb{L}}$-chamber structure on the positive cone $\mathcal{P}_{\mathbb{L}}$ of the ambient lattice $\mathbb{L}$ into which is assumed to be embedded in S.
- As described in section 1.2, we use the $\mathcal{P}_{\mathbb{L}}$-chamber structure to induce a $\mathcal{P}_S$-chamber structure on the positive cone $\mathcal{P}_S$ of S.

In this section and until the remainder of the first part of our thesis, we will present Borcherds' method and explain how we implemented it. We proceed by using the fundamental building blocks provided by Shimada in his article [19] as a basis and present all the details and developments which have been obtained during our study.

Borcherds' method is an algorithmic process that produces a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ by exploring and processing the $\mathcal{P}_S$-chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ until a complete set of representatives of $\mathbf{H}$-congruence classes of $\mathcal{P}_S$-chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ has been obtained.

Our approach can be decomposed along three axes:

- We start by studying the portion of the $\mathcal{P}_S$-chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$. This structure is a theater where a good part of our story unfolds. It is therefore crucial that we have a clear vision of this portion of the chamber structure.
- We introduce the procedures used by Borcherds' method to **explore** the portion of the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$.
- We introduce the tools that enable Borcherds' method to **process** this portion of the chamber structure.

We will conclude with a figure which sums up everything regarding Borcherds' method. We provide our ready-to-use implementation of Borcherds' method with multi-core support on our website K3surfaces.com. We used **Pool** from the Python multiprocessing library to make use of process-based parallelism in our implementation of the method.

Chamber structure over Nef$(X) \cap \mathcal{P}_S$

The first fact of importance which should be exhibited is that $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is tiled by chambers of the induced $\mathcal{P}_S$-chamber structure.

To see this, we first have to recall that we have seen in section 1.2 that the walls of the $\mathcal{P}_S$-chambers structure all arise by taking the orthogonal complement in $\mathcal{P}_S$ of elements of the set

$$\mathcal{R}_{\mathbb{L}|S} = \{x_S \in S^\vee \mid x \in \mathcal{R}_{\mathbb{L}}, \langle x_S, x_S \rangle_{S^\vee} < 0\}\,.$$

Note that any $x \in S \subset S^\vee$ satisfying

$$\langle x, x \rangle_S = -2$$

also satisfies $x \in \mathcal{R}_{\mathbb{L}|S}$. A fact of importance for the remainder of this section is that this statement also holds for classes of divisors of curves playing a central role on $K3$ surfaces: Classes of divisors associated with smooth rational curves, also known as classes of (-2)-curves, or as (-2)-curves. Thus, each class of a smooth rational curve can be associated with a wall of some chamber of the $\mathcal{P}_S$-chamber structure. Moreover, a classical result which can be found in Huybrechts' book [5] states that each class of a smooth rational curve can be associated with a wall of $\mathrm{Amp}(X)$. Keeping in mind that $\mathrm{Amp}(X)$ and $\mathrm{Nef}(X)$ are related by the equality

$$\mathrm{Amp}(X) = \mathrm{Int}(\mathrm{Nef}(X)),$$

we deduce that no (-2)-curve is superfluous for defining a wall of $\mathrm{Nef}(X)$.

What about $\mathrm{Nef}(X) \cap \mathcal{P}_S$?

The answer is provided by a useful result from Huybrechts' book [5] with the following characterization of the boundary of $\mathrm{Nef}(X) \cap \mathcal{P}_S$.

A class $C \in S$ belonging to the boundary of $\mathrm{Nef}(X)$ satisfies either one of the two following properties:

- The equality $C^2 = 0$ holds.
- There exists a class E of a smooth rational curve such that $\langle C, E \rangle_S = 0$.

Since all classes in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ have a strictly positive self-intersection, we deduce that each (-2)-curve on X can be associated with a wall of $\mathrm{Nef}(X) \cap \mathcal{P}_S$. Such walls are called (-2)-walls, and bound $\mathrm{Nef}(X) \cap \mathcal{P}_S$. The induced $\mathcal{P}_S$-chamber structure thus contains a natural chamber substructure covering $\mathrm{Nef}(X) \cap \mathcal{P}_S$, and bound by (-2)-walls. Not crossing these walls is a golden rule that Borcherds' method must follow. Indeed, the method would otherwise leave its work area over $\mathrm{Nef}(X) \cap \mathcal{P}_S$, thus potentially distorting the data and results obtained. The procedure **RatDetect** detailed in section 1.7.1 is capable of detecting (-2)-walls. This procedure can be viewed as a compass that allows the method not to get lost during its journey.

Exploring the chamber structure

Borcherds' method pursues the exploration of the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ by moving from chamber to chamber. In order to formalize the movement of Borcherds' method, we first have to introduce the notion of adjacency for chambers. Let D, D' be two $\mathcal{P}_S$-chambers having the property of sharing a wall $(v)^\perp$ with $v \in S \otimes \mathbb{R}$.

Definition 26. We say that D and D' are *adjacent* along the wall $(v)^\perp$ whenever the intersection

$$D \cap D' \cap (v)^\perp$$

contains a non-empty open subset of $(v)^\perp$. We also say that the chamber D (resp. D') is adjacent to D' (resp. D) along the wall $(v)^\perp$.

Using a chamber $D_0 \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$ as a reference point, the notion of adjacency is used to layer the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ into various

levels. The chamber D_0 will often be referred to as the *initial chamber*.

Definition 27. The notion of *level* is defined iteratively:

- The initial chamber D_0 is the only level 0 chamber.
- A chamber adjacent to a level $l-1$ chamber but not adjacent to a level $l-2$ chamber is said to be of level l.

The notion of level enables us to give a precise characterization of our object of study: Starting from an initial chamber D_0 contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$, Borcherds' method is an iterative process that explores and processes the chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$, level by level, until a complete set of representatives of **H**-congruence classes of chambers has been produced. In order to navigate within the chamber structure on $\mathrm{Nef}(X) \cap \mathcal{P}_S$, Borcherds' method must possess the three following features:

- Borcherds' method must be able to move from chamber to chamber.

To this end, Borcherds' method leans on the procedure **WeylAdj** presented in section 1.7.2. Given the Weyl vector w of a chamber D and the data of an element $v \in S^\vee$ such that $(v)^\perp$ is a wall of D, the procedure **WeylAdj** computes the Weyl vector w' of the chamber D' adjacent to D along $(v)^\perp$.

- Borcherds' method must possess the ability to detect (-2)-walls, that is, walls $(v)^\perp$ where v satisfies $\langle v, v\rangle_S = -2$ and $v \in S$. Doing so is the purpose of the procedure **RatDetect**, from section 1.7.1, which takes as input an element $v \in S^\vee$ and determines whether $(v)^\perp$ is a (-2)-wall.

Indeed, we have seen that the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is bounded by (-2)-walls. Hence, in case the method crosses a (-2)-wall, it leaves the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$. Crossing such walls must be avoided at all costs. We recall that the set of walls of a $\mathcal{P}_S$-chamber D is denoted by $\Omega(D)$ and contained in the set $\mathcal{R}_{\mathbb{L}|S}$ of elements of $S \otimes \mathbb{Q}$ having negative self-intersection.

- The third desired feature of the method is that it should never backtrack.

Assume that D is a chamber of level k, and that Borcherds' method is currently exploring the adjacencies around D. Then, the method should not be allowed to explore adjacencies along walls of D leading to chambers of level $k-1$. These chambers have indeed already been explored and processed during previous iterations. The method thus also needs an anti-backtracking procedure to immediately recognize the walls of a given chamber leading to a chamber of lower level. We use the notation

$$\overline{\Omega}(D)$$

to denote the set of walls of D from which have been removed the walls leading to chambers of level $k-1$. Explanations regarding our approach to determine $\overline{\Omega}(D)$ can be found by clicking here. This set will often be referred to (in particular, on figures) as the set of walls of D with respect to anti-backtracking.

1.7.1 Procedure RatDetect

This section is based on Shimada's guidelines which can be found in point 2.2 of Algorithm 6.1 from his article [19]. Let D be a $\mathcal{P}_S$-chamber. Determining whether the wall $(v)^\perp$ associated with an element $v \in \Omega(D)$ is a (-2)-wall amounts to:

- **Step n°1 -** Determining the integer solution set $\mathscr{S}_v$ of the equation

$$x^2 \langle v, v \rangle_{S^\vee} = -2$$

  of the variable $x \in \mathbb{Z}$.

- **Step n°2 -** If $\mathscr{S}_v = \emptyset$, then $(v)^\perp$ is not a (-2)-wall. Otherwise, we check whether there exists an element $q \in \mathscr{S}_v$ such that $qv \in S$. If this is the case, then $(v)^\perp$ is a (-2)-wall. Otherwise, $(v)^\perp$ is not a (-2)-wall.

Accomplishing the task of **Step n°1** should not present any difficulty. In order to deal with **Step n°2**, assume that $\mathscr{S}_v \neq \emptyset$ an let $q \in \mathscr{S}_v$. Proceeding as described

in section 1.3, we compute the image of qv in $\mathbb{L} \otimes \mathbb{R}$ and project it onto $S \otimes \mathbb{R}$. If the resulting vector has integer coordinates, then it belongs to S.

1.7.2 Procedure WeylAdj

Given an element $v \in \Omega(D)$ and the Weyl vector w of $\mathcal{P}_S$-chamber D, the algorithms 5.13 and 5.14 outlined in Shimada's article [19] can be used to compute the Weyl vector $w_{D'}$ of a $\mathcal{P}_S$-chamber D' adjacent to D along the wall $(v)^\perp$. We combined both of these algorithms into a single procedure: The procedure **WeylAdj** takes as input an element $v \in \Omega(D)$ and the Weyl vector w of a $\mathcal{P}_S$-chamber D and outputs the Weyl vector w' of the $\mathcal{P}_S$-chamber D' adjacent to D along the wall $(v)^\perp$. We begin by stating Shimada's algorithms in a user-friendly form, and adopt a step-by-step approach. Doing so enables us to provide as many details as possible, thus enabling our readers to easily implement their own versions of this important building block of Borcherds' method. We now present Shimada's procedure to compute the Weyl vector the chamber D' adjacent to D along $(v)^\perp$ where

$$v \in \Omega(D) \subset \{v \in S \otimes \mathbb{Q} \mid \langle v, v \rangle_S < 0\}.$$

In order to compute a Weyl vector w' of D', proceed as follows:

- **Step n°1:** Compute the set

$$P_v = \left\{ r \in \mathcal{R}_{\mathbb{L}} \mid (v)^\perp \subset (r)^\perp \right\}.$$

- **Step n°2:** Choose a complete set of representatives

$$P'_v = \{r_1, \ldots, r_N\}$$

of $P_v \,/\, \{\pm 1\}$.

- **Step n°3:** Choose an element $u \in \mathbb{L} \otimes \mathbb{Q}$ such that

$$i \neq j \quad \Longrightarrow \quad \frac{\langle u, r_i \rangle_{\mathbb{L}}}{\langle w, r_i \rangle_{\mathbb{L}}} \neq \frac{\langle u, r_j \rangle_{\mathbb{L}}}{\langle w, r_j \rangle_{\mathbb{L}}}.$$

  and sort the elements of P'_v in such a way that

$$i < j \quad \Longrightarrow \quad \frac{\langle u, r_i \rangle_{\mathbb{L}}}{\langle w, r_i \rangle_{\mathbb{L}}} < \frac{\langle u, r_j \rangle_{\mathbb{L}}}{\langle w, r_j \rangle_{\mathbb{L}}}$$

  holds for all $r_i, r_j \in P'_v$.

- **Step n°4:** Denote by $s_i \in O^+(\mathbb{L})$ the reflection with respect to r_i. Then

$$w^{s_1 s_2 \cdots s_N} := (s_1 \circ s_2 \circ \cdots \circ s_N)(w)$$

  is a Weyl vector of D'. Note that a proof is given in [19, section 5].

We explain how we implemented Shimada's algorithm, step-by-step. Before proceeding further, recall that given an element $v \in S \otimes \mathbb{R}$, we define

$$(v)^{\perp} = \{x \in S \otimes \mathbb{R} \mid \langle x, v \rangle_S = 0\} \cap \mathcal{P}_S.$$

Step n°1 - Consider the subspace

$$\mathcal{V} = \mathbb{R}v \oplus (R \otimes \mathbb{R})$$

of $\mathbb{L} \otimes \mathbb{R}$. We denote by $\mathrm{pr}_{\mathcal{V}}(r)$ the projection onto $\mathcal{V}$ of an element $r \in \mathbb{L}$. Note that the set

$$P_v = \left\{ r \in \mathcal{R}_{\mathbb{L}} \mid (v)^{\perp} \subset (r)^{\perp} \right\}$$

can be expressed as

$$P_v = \{r \in \mathcal{R}_{\mathbb{L}} \mid r_S \in \mathbb{R}v\}$$

where r_S denote the orthogonal projection onto $S^{\vee}$ of an element $r \in \mathbb{L}$.

Note that since $v \in S^\vee$ and $r \in \mathbb{L}$, taking the inclusion

$$(v)^\perp \subset (r)^\perp$$

only makes sense if we view $(v)^\perp$, which has been initially defined as a hyperplane of $\mathcal{P}_S$, as a hyperplane of $\mathcal{P}_\mathbb{L}$. The assumption that S is embedded primitively into $\mathbb{L}$ in such a way that $\mathcal{P}_S \subset \mathcal{P}_\mathbb{L}$ enables us to do so. We explain how to compute the set P_v explicitly. Doing so is the exclusive purpose of Shimada's Algorithm 5.13. We follow his guidelines and provide all the necessary additional details. Shimada starts by defining an initially empty set $P = \{\,\}$ and computes the set

$$\mathscr{S} = \left\{\alpha \in \mathbb{Q} \mid \alpha v \in S^\vee,\, \alpha^2 v^2 \geq -2\right\}.$$

In order to explicitly determine this set, we proceeded as follows: Assume that $\alpha \in \mathscr{S}$. Since α is by definition a rational, we express it as

$$\alpha = p/q$$

with $p, q \in \mathbb{Z}$, and $q \neq 0$. Denote by

$$\left\{s_1^\vee, \ldots, s_\rho^\vee\right\}$$

a basis for $S^\vee$ (see the Toolbox section 1.3 for guidelines on the choice of a basis for $S^\vee$) and express the element $v \in \Omega(D) \subset S^\vee$ in terms of its coordinates $v_i \in \mathbb{Z}$ for $1 \leq i \leq \rho$ with respect to this basis, so that

$$v = v_1 s_1^\vee + \cdots + v_\rho s_\rho^\vee.$$

Let us take apart the defining conditions of the set $\mathscr{S}$. We have $p/q \in \mathscr{S}$ if and only if the two following conditions are satisfied:

- The element αv, i.e., $(p/q)v$, must belong to $S^\vee$. This important requirement can only be fulfilled if the integer q divides each of the coordinates

v_i of v. That is,

$$q \mid v_i$$

must be true for $1 \leq i \leq \rho$.

We thus introduce the set

$$\mathscr{S}_0 = \{n \in \mathbb{Z} \ \mid \ n \mid v_1, \ldots, n \mid v_\rho\}$$

of all integers satisfying this property. Doing so enables us to know all possible denominators q for p/q.

- The condition $\alpha^2 v^2 \geq -2$ must hold.

For each $q \in \mathscr{S}_0$ we thus solve for x the inequality

$$x^2 \langle v, v \rangle_{S^\vee} \geq -2q^2$$

and store the solutions, when such solutions exist, into a set, say, an initially empty set $\mathscr{S}_1$. It is then clear that

$$\begin{aligned} \mathscr{S} &= \left\{\alpha \in \mathbb{Q} \mid \alpha v \in S^\vee,\, \alpha^2 v^2 \geq -2\right\} \\ &= \{p/q \in \mathbb{Q} \mid \ p \in \mathscr{S}_1, q \in \mathscr{S}_0\}\,. \end{aligned}$$

In order to explicitly compute the set $\mathscr{S}_0$, we proceed as follows: Let v_{max} be the largest (in absolute value) of the cordinates of $v \in S^\vee$. Define

$$\mathcal{T} = \{-v_{\text{max}}\,, -v_{\text{max}} + 1, \ldots, v_{\text{max}} - 1, v_{\text{max}}\} \subset \mathbb{Z}.$$

The set $\mathscr{S}_0$ can then finally be obtained as

$$\mathscr{S}_0 = \{m \in \mathcal{T} \mid m \text{ divides } v_i \ \text{ for } \ 1 \leq i \leq \rho\}\,,$$

which can be easily computed. Note that if we follow the guidelines available in

sections 1.5 and 1.5.2 to compute the set of walls of a chamber, we always obtain

$$\mathscr{S}_0 = \{\pm 1\}$$

no matter which element $v \in \Omega(D)$ or which $\mathcal{P}_S$-chamber D is used. Indeed, it follows from the directives contained in these sections that the coordinates $v_1, \ldots, v_\rho$ of elements $v \in S \otimes \mathbb{Q}$ inducing walls must satisfy

$$\gcd(v_1, \ldots, v_\rho) = 1.$$

We now compute $\mathscr{S}_1$. To this end, initially define it as an empty set $\mathscr{S}_1 = \{\,\}$ and proceed as follows: For each $q \in \mathscr{S}_0$, solve

$$x^2 v^2 \leq -2q^2$$

for $x \in \mathbb{Z}$ and store the resulting solutions into the set $\mathscr{S}_1$. The desired set $\mathscr{S}$ is then finally be obtained as

$$\mathscr{S} = \{p/q \in \mathbb{Q} \mid \; p \in \mathscr{S}_1, q \in \mathscr{S}_0\}\,.$$

For each $\alpha \in \mathscr{S}$ we then compute

$$c_\alpha = -2 - \alpha^2 v^2$$

and let

$$c_{\max} = \max_{\alpha \in \mathscr{S}}(c_\alpha).$$

Recall that we denote by $R = S^\perp$ the orthogonal complement of S in $\mathbb{L}$. Since $R^\vee$ is negative definite, we can make use of a short lattice vectors enumeration algorithm to compute the set

$$\{x \in R^\vee \mid \langle x, x\rangle_{R^\vee} \leq c_{max}\}\,.$$

Knowledge of this set enables us to obtain a set

$$R^\vee[c_\alpha] = \{x \in R^\vee \mid \langle x, x\rangle_{R^\vee} = c_\alpha\}$$

for each $\alpha \in \mathscr{S}$. We now have all the necessary ingredients to determine

$$P_v = \left\{r \in \mathcal{R}_{\mathbb{L}} \mid (v)^\perp \subset (r)^\perp\right\}.$$

For each $\alpha \in \mathscr{S}$ and $u \in R^\vee[c_\alpha]$, determine whether $\alpha v + u$ belongs to $\mathbb{L}$. To this end, we use our knowledge of bases of $S^\vee$ and $R^\vee$ made of elements of $\mathbb{L}$ to express both $v \in S^\vee$ and $u \in R^\vee$ as elements of $\mathbb{L} \otimes \mathbb{R}$. If the sum $\alpha v + v$ belongs to $\mathbb{L}$, i.e., has integer coordinates with respect to the standard basis of $\mathbb{L}$, append $\alpha v + u$ to P_v. This is thus how the set P_v can be computed.

Step n°2 - We then have to compute a complete set of representatives of $P_v / \pm 1$. Create an initially empty set P'_v and proceed as follows: For each $q \in P_v$, if

$$-q \notin P'_v$$

then append q to P'_v. Assume that the resulting set has cardinality N for some positive integer N and is expressed as:

$$P'_v = \{r_1, \ldots, r_N\} \subset P_v$$

Step n°3 - We then have to pick an element $u \in \mathbb{L} \otimes \mathbb{Q}$ such that $i \neq j$ implies

$$\langle u, r_i\rangle_{\mathbb{L}} / \langle w, r_i\rangle_{\mathbb{L}} \neq \langle u, r_j\rangle_{\mathbb{L}} / \langle w, r_j\rangle_{\mathbb{L}}.$$

This can be done in two ways:

- By randomly generating an element of $u \in \mathbb{L} \otimes \mathbb{Q}$ until the condition

$$\left\langle u, r_i - \frac{\langle w, r_i\rangle_{\mathbb{L}}}{\langle w, r_j\rangle_{\mathbb{L}}} r_j \right\rangle \neq 0 \tag{1.20}$$

is fullfilled for $1 \leq i, j \leq N$.

- The other way we offer may necessite less attempts to form the set

$$\mathcal{P} = \{(r_i, r_j) \mid r_i, r_j \in P'_v,\ i < j\}\,.$$

We use the notation $\left(p^{(1)}, p^{(2)}\right)$ to denote elements $p \in \mathcal{P}$. In practice, the element

$$u = r_0 + \sum_{i=1}^{\mathrm{Card}(\mathscr{P})} p_i^{(1)} - \sum_{p \in \mathcal{P}} \frac{\left\langle w, p^{(1)} \right\rangle_{\mathbb{L}}}{\left\langle w, p^{(2)} \right\rangle_{\mathbb{L}}} p_i^{(2)},$$

where r_0 is a randomly generated element of $\mathbb{L}$, may satisfy the inequalities (1.20).

If this is not the case, add another randomly generated element r'_0 of $\mathbb{L}$ to u and determine whether the resulting element $u_{\text{upd}} = u + r'_0$ thus obtained satisfies the inequalities (1.20). Repeat until these inequalities are satisfied.

Step n°4 - Assume that a suitable element $u \in \mathbb{L} \otimes \mathbb{Q}$ has been obtained. Shimada then re-labels the elements of P'_v according to the following rule: If the indices of re-labelled elements $r_i, r_j \in P'_v$ satisfy $i < j$ then the inequality

$$\frac{\langle u, r_i \rangle_{\mathbb{L}}}{\langle w, r_i \rangle_{\mathbb{L}}} < \frac{\langle u, r_j \rangle_{\mathbb{L}}}{\langle w, r_j \rangle_{\mathbb{L}}}$$

must hold. Denote by s_i the be reflection

$$s_i : x \longmapsto x + \langle x, r_i \rangle_{\mathbb{L}}\, r_i$$

associated with an element $r_i \in P_v$. The Weyl vector $w_{D'}$ of the chamber adjacent to D along $(v)^{\perp}$ can then be obtained from w_D as

$$w_{D'} = (s_1 \circ \cdots \circ s_N)\,(w_D)\,.$$

Processing the chamber structure

We introduced the tools which enable Borcherds' method to progress within the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$. We now introduce the tools which enable Borcherds' method to process the portion of this chamber structure it explores, and thus accomplish its purpose: Computing a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$. Before proceeding further, let us review the notational conventions that will be used regarding transformations of $O(S)$: We consider that elements $g \in O(S)$ act on elements of S and $S \otimes \mathbb{Q}$ from the right. That is, the image of an element $b \in S$ under the action of an element $g \in O(S)$ is denoted by bg, or by b^g. Similarly, we denote by

$$D^g = \{b^g \mid b \in D\}$$

the image of a $\mathcal{P}_S$-chamber under the action of an element $g \in O(S)$. Borcherds' method enforces two courses of action in order to exhibit generators $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$:

- For each $\mathcal{P}_S$-chamber $D \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$ it explores, the method can take advantage of the fact that generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ can be obtained by computing a generating set of

  $$\mathrm{Aut}_{\mathbf{H}}(D) = \{g \in \mathbf{H} \mid D = D^g\},$$

  which, as established by Shimada in [19], is a finite subgroup of

  $$\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S).$$

From the input data of the set $\Omega(D)$ of walls of a $\mathcal{P}_S$-chamber $D \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$, the procedure **AutChamber**, which is based on Shimada's Algorithm 3.18 from [19], is introduced in section 1.7.3 and computes a generating set of $\mathrm{Aut}_{\mathbf{H}}(D)$.

- The main course of action followed by Borcherds' method to produce generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ is based on the method's capability to identify relations of $\mathbf{H}$-congruency between $\mathcal{P}_S$-chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$.

The relation of $\mathbf{H}$-congruency between chambers will be central for the rest of our study, and is defined as follows:

Definition 28. Two $\mathcal{P}_S$-chambers D and D' contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ are said to be $\mathbf{H}$-congruent whenever there exists an isometry of $\mathbf{H}$ sending either one of D or D' onto the other.

That is, we say that D and D' are $\mathbf{H}$-congruent if there exists an element $g \in \mathbf{H}$ such that $D' = D^g$. When this is the case, the chambers D and D' both belong to the same $\mathbf{H}$-congruence class of chambers. The procedure **CongChecker**, based on Shimada's Algorithm 3.19 from [19] and described in section 1.7.4, takes as input the respective sets of walls $\Omega(D)$ and $\Omega(D')$ of $\mathcal{P}_S$-chambers $D, D' \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$ and determines whether these chambers belong to the same $\mathbf{H}$-congruence class. When this is the case, the procedure **CongChecker** outputs at least one transformation $g \in \mathbf{H}$ such that $D' = D^g$. Both procedures **AutChamber** and **CongChecker** are based on the same underlying mechanics. As indicated by Shimada in his article, the latter are ultimately brute force flavored. Note that massive gains can be realized when repeated use of **CongChecker** is done using process-based parallelism. Our Python implementation of Borcherds' method uses the **Pool** object from the Python **multiprocessing** library and can thus take advantage of the multi-core architecture of a CPU. We provide more details about this matter in section 1.11.1, the *Poolized Borcherds' method*. Shimada's Algorithm 3.18 from [19], on which is based our implementation of **AutChamber**, relies on the fact that having knowledge of the set $\Omega(D)$ of walls of a $\mathcal{P}_S$-chamber D is enough to precisely define the domain of possibilities in terms of the generators of $\mathrm{Aut}_{\mathbf{H}}(D)$. In his article [19], Shimada indeed states that such transformations can be characterized by the fact that they must belong to $\mathbf{H}$ and above all must act as permutations of $\Omega(D)$. Note that an ad-

ditional development brought by this thesis is that a generalized membership criterion for $\mathbf{H}$ is provided in section 1.6. From the input data of $\Omega(D)$, the procedure **AutChamber** thus generates all possible transformations acting as permutations of $\Omega(D)$ and then tests each of them for membership in $\mathbf{H}$ by enforcing the membership criterion given in the proposition 24 from section 1.6. This procedure thus enables Borcherds' method to obtain a generating set of $\mathrm{Aut}_{\mathbf{H}}(D)$ for any $\mathcal{P}_S$-chamber $D \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$ it explores. The procedure **CongChecker** is based on analogous principles. As demonstrated by Shimada, knowledge of the walls of $\mathcal{P}_S$-chambers D and D' is enough to precisely define the domain of possibilities in terms of isometries sending D onto D'. Such transformations are characterized by the fact that they must establish a bijection between $\Omega(D)$ and $\Omega(D')$ while also being elements of $\mathbf{H}$. From the input data of $\Omega(D)$ and $\Omega(D')$, the procedure **CongChecker** generates all possible transformations which could send $\Omega(D)$ onto $\Omega(D')$, and then enforces the membership criterion for $\mathbf{H}$ in order to single out the elements sending D onto D'. Note that in case sets of walls of the same chambers is are as input into the procedure **CongChecker**, the latter will behave exactly like the procedure **AutChamber** and output a generating set of $\mathrm{Aut}_{\mathbf{H}}(D)$. Both of these procedures could not exist without the following proposition established by Shimada in [19]:

Proposition 29. *Any defining set Δ of a $\mathcal{P}_S$-chamber D spans $S \otimes \mathbb{R}$.*

We have seen in section 1.5 that the set $\Omega(D)$ of walls of a $\mathcal{P}_S$-chamber D, which is called the primitively minimal defining set of D by Shimada is by definition a defining set of the chamber D. Proposition 29 hence implies that the cardinality of the set of walls $\Omega(D)$ of a $\mathcal{P}_S$-chamber D is at least equal to the Picard number of the $K3$ surface under study. We thus form the set

$$\mathrm{Tups}(\Omega(D)) = \{(m_1, m_2, \ldots, m_\rho) \mid m_i \in \Omega(D), 1 \leq i \leq \rho_X\}$$

of ρ-sized tuples of elements of $\Omega(D) \subset S^\vee$, from which can be picked a tuple

$$\tau_{\text{gen}} \in \text{Tups}(\Omega(D))$$

having the property of being made of elements which span $S \otimes \mathbb{R}$. Such a tuple τ_{gen} is called a generating tuple. Finding such tuples is the purpose of the procedure **GentTup**.

Procedure GentTup: Assume given as input the set of walls $\Omega(D)$ of a $\mathcal{P}_S$-chamber D. Compute the set $\text{Tups}(\Omega(D))$. For each $\tau = (m_1, m_2, \ldots, m_\rho)$ in $\text{Tups}(\Omega(D))$, form the $(\rho \times \rho)$-sized matrix obtained by taking as columns the elements of τ and compute its determinant. If the latter is non-zero, then τ is a generating tuple. Otherwise, τ is not a generating tuple. Shimada's proposition 29 ensures that it is always possible to determine a generating tuple. As soon as a tuple with this property, i.e., a generating tuple, is found, the procedure **GentTuple** outputs it as the generating tuple.

Assume that a generating tuple τ_{gen} of either D or D' has thus been obtained, say, a generating tuple of D. We now introduce the procedure **TupLink**, which is intended to:

- Enable **AutChamber** to determine transformations which act as a permutation of the set of walls of a chamber.
- Enable **CongChecker** to determine transformations sending the set of walls $\Omega(D)$ of a $\mathcal{P}_S$-chamber D onto the set of walls $\Omega(D')$ of another $\mathcal{P}_S$-chamber D'.

Given a generating tuple $\tau_{\text{gen}} \in \Omega(D)$ and a tuple $\tau \in \text{Tups}(\Omega(D'))$, the procedure **TupLink** attempts to produce a $(\rho \times \rho)$-sized matrix $M_{\tau, \tau_{\text{gen}}}$ sending τ onto τ_{gen}, where $\rho = \text{rank}(S)$, thus trying to link these tuples, as follows:

Procedure TupLink: Assume given tuples

$$\tau_1 = (t_1, \ldots, t_\rho) \qquad \text{and} \qquad \tau_2 = (v_1, \ldots, v_\rho)$$

with $t_i, v_i \in S^\vee$ for $1 \leq i \leq \rho$. Assume moreover that either one of τ_1, τ_2 is a generating tuple. For example, assume that τ_2 is a generating tuple, i.e., that its elements are linearly independent. Our aim consists in determining an invertible $(\rho \times \rho)$-sized matrix M_{τ_1,τ_2} satisfying

$$M_{\tau_1,\tau_2} t_i = v_i \qquad \text{for} \qquad 1 \leq i \leq \rho. \tag{1.21}$$

To this end, we proceed as follows: Let A be the $(\rho \times \rho)$-sized matrix formed by taking the elements of τ_1 as columns, that is,

$$A = \begin{pmatrix} t_1 & | & t_2 & | & \cdots & | & t_{\rho-1} & | & t_\rho \end{pmatrix}.$$

and denote by B the $(\rho \times \rho)$-sized matrix obtained by taking the elements of τ_2 as columns that, is,

$$B = \begin{pmatrix} v_1 & | & v_2 & | & \cdots & | & v_{\rho-1} & | & v_\rho \end{pmatrix}.$$

Note that our assumption on the linear independence of the elements of τ_2 enables us to assert that the matrix B is invertible. We then determine whether

$$M_{\tau_1,\tau_2} = AB^{-1}$$

establishes a one-to-one correspondence between the elements of τ_1 and τ_2, i.e., satisfies the equalities resulting from expression (1.21). When this is the case, output M_{τ_1,τ_2}. We have to take into account the fact that whenever M_{τ_1,τ_2} is expected to be invertible, then the matrix A must also be invertible.

This can only happen if τ_1 is a generating tuple. We thus have to keep in mind that whenever the procedure **TupLink** is applied with the hope of obtaining invertible transformations, both tuples used as input data should be generating tuples. Time and resources would otherwise be wasted. We denote by

$$\text{Tups}_{\text{gen}}(\Omega(D)) \subseteq \text{Tups}(\Omega(D))$$

the set made of all the generating tuples contained in $\text{Tups}(\Omega(D))$, which can thus be obtained by testing each tuple with **GentTup**. We have all the tools required to formalize the procedures **AutChamber** and **CongChecker**.

1.7.3 Procedure AutChamber

This procedure, based on Shimada's algorithm 3.18, takes as input the set of walls $\Omega(D)$ of a $\mathcal{P}_S$-chamber D and outputs a generating set of $\text{Aut}_{\mathbf{H}}(D)$. Define an initially empty set $\mathcal{A} = \{\}$. Apply the procedure **GenTup** each element of $\text{Tups}(\Omega(D))$, in order to obtain the set $\text{Tups}_{\text{gen}}(\Omega(D))$. Fix a generating tuple $\tau_{\text{gen}} \in \text{Tups}_{\text{gen}}(\Omega(D))$. For each generating tuple $\tau \neq \tau_{\text{gen}}$, use the procedure **TupLink** to determine whether there exist $(\rho \times \rho)$-sized matrices M sending the set of elements of τ onto the set of elements of τ_{gen}. When this is the case, proceed as follows for each such matrix M thus obtained:

- Determine whether all the entries of the matrix M are integers. When this is not the case, discard M.
- Determine whether M acts as a permutation on the elements of $\Omega(D)$. That is, determine whether the image of the set $\Omega(D)$ under the matrix transformation M coincides with $\Omega(D)$ itself. Discard M if it does not fulfill this requirement.
- When M acts as a permutation of $\Omega(D)$, apply the procedure **MemberCrit** to $M_{\tau,\tau_{\text{gen}}}$ in order to determine whether it belongs to $\mathbf{H}$. When this is the case, append M to the set $\mathcal{A}$.

The article [19] from Shimada then ensures that the resulting set $\mathcal{A}$ obtained at the end of the procedure satisfies

$$\mathcal{A} = \mathrm{Aut}_{\mathbf{H}}(D).$$

Note that if $\mathcal{A}$ is empty then

$$\mathrm{Aut}_{\mathbf{H}}(D) = \{\mathrm{Id}\}\,.$$

1.7.4 Procedure CongChecker

The Procedure **CongChecker** is based on Shimada's Algorithm 3.19 from his article [19] and relies on the same mechanics than its sister procedure **AutChamber**. The congruence testing procedure takes as input the data of sets of walls $\Omega(D)$ and $\Omega(D')$ of $\mathcal{P}_S$-chambers D and D' and determines whether the latter are $\mathbf{H}$-congruent by proceeding as follows: Define an initially empty set $\mathcal{A} = \{\,\}$. Apply the procedure **GentTup** to each element of $\mathrm{Tups}(\Omega(D'))$ until a generating tuple $\tau_{\mathrm{gen}} \in \mathrm{Tups}(\Omega(D'))$ is obtained. Note that proposition 29 guarantees that obtaining such a tuple is always possible. Compute the set $\mathrm{Tups}_{\mathrm{gen}}(\Omega(D))$ of all the generating tuples contained in $\mathrm{Tups}(\Omega(D))$ by applying **GentTuple** to each element of the latter. Proposition 29 ensures that this set will contain at least one element. For each $\tau \in \mathrm{Tups}_{\mathrm{gen}}(\Omega(D))$, apply the procedure **TupLink** in order to determine whether there exists at least one matrix $(\rho \times \rho)$-sized matrix $M_{\tau,\tau_{\mathrm{gen}}}$ sending the set of elements of τ onto the set of elements of τ_{gen}. If all the coefficients of $M_{\tau,\tau_{\mathrm{gen}}}$ are integers and this matrix moreover establishes a one-to-one correspondence between $\Omega(D)$ and $\Omega(D')$, use the procedure **MemberCrit** to check whether $M_{\tau,\tau_{\mathrm{gen}}}$ belongs to $\mathbf{H}$. If this is the case, append $M_{\tau,\tau_{\mathrm{gen}}}$ to $\mathcal{A}$. When all tuples $\tau \in \mathrm{Tups}(\Omega(D'))$ have been processed, output the set $\mathcal{A}$. At the end of the process, if $\mathcal{A}$ is non-empty and contains at at least one non-trivial element then **CongChecker** returns a boolean value of **True** with the data of the elements of $\mathcal{A}$. Such elements thus establish that the $\mathcal{P}_S$-chambers D and D' belong to the same $\mathbf{H}$-congruence class of

chambers. Otherwise, **CongChecker** outputs a boolean value of **False**. More details about the way we implemented Shimada's congruence testing procedure can be found by clicking here. Note that developments obtained during this thesis resulted in huge enhancements to Shimada's congruence testing procedure, which has been detailed in [19] almost a decade ago. With efficiency and parallel deployment in mind, we explain on the online support dedicated to this thesis how our approach to congruence testing enabled us to obtain fantastic performance gains. We provide a concrete example where a given chamber had to be tested against 80231 other chambers for congruency. New criteria for congruency combined with parallel deployment enabled us to divide the total computation time for these 80231 tests by 1000 (conservative estimate) compared to the times measured when the 2013 approach from [19] is used to the letter. Click here to access an online section in which are detailed the developments on congruence testing obtained during this thesis.

1.7.5 Borcherds' method

We now possess all the tools required in order to introduce Borcherds' method itself. In this section, we will proceed as follows:

- We start by making a precise survey of the framework required in order to successfully execute Borcherds' method and obtain a generating set of the automorphism group a complex $K3$ surface.
- We then explain in terms of tuples and sets how we formalized the data of chambers, which are undeniably objects of paramount importance within Borcherds' method.
- Using Shimada's take on Borcherds' method from his 2013 article [19], we then describe how we put together the building blocks that have been introduced so far to successfully implement Borcherds' method. We also describe all the evolutions, improvements and developments obtained during this thesis.

The reader should note that we provide our ready-to-use implementation of Borcherds' method for complex algebraic $K3$ surfaces on K3surfaces.com. An online section of this thesis also provides a variety of step-by-step examples of applications of Borcherds' method. These examples show how a computer-based algorithmic approach can lead to a wealth of concrete information and results on classical cases, originally obtained by hand when published decades ago. As far as we know, we also provide concrete answers to questions that had been open for many years, in some of these step-by-step examples. Techniques illustrated through these examples can then be used to study other surfaces.

Let X be a $K3$ surface. The input required in order to use our fully automated implementation of Borcherds' method consists of

- The data of elements $v_1, \ldots, v_\rho \in \mathbb{L}$ such that the map

$$\iota : [x_1, \ldots, x_\rho]_S \longmapsto x_1 v_1 + \cdots + x_\rho v_\rho$$

  is a primitive embedding of $S = \mathrm{NS}(X)$ into one of the three even hyperbolic lattices $\mathbb{L}$ mentioned in section 1.1.2 and chosen depending on the Picard number of X. Click here for more details on this matter.

- A Gram matrix G_S of S.

- An ample class $a_0 \in S$ that will be used to update the embedding of S into $\mathbb{L}$, if necessary.

Before proceeding further, we have to indicate that we choose to refer to Borcherds' method as if it was a system embodied by a small animal obeying certain rules and capable of making decisions within a predefined framework. Note that we use a hamster emoji in many figures, and that this hamster is meant to embody Borcherds' method. Doing so enables us to illustrate the fundamental concepts, principles, and mechanics behind the method in a simple and accessible way, without ever violating the underlying theory.

For a better understanding of the material presented in this section, it is important to remember the purpose of our object of study: In the framework of complex $K3$ surfaces, Borcherds' method produces a generating set of

$$\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S) = \{g \in \mathbf{H} \mid \forall x \in \mathrm{Nef}(X) \cap \mathcal{P}_S, gx \in \mathrm{Nef}(X) \cap \mathcal{P}_S\}$$

where we recall that $\mathbf{H}$ is a subgroup of $O^+(S)$ that can be explicitly characterized by a generalized membership criterion, provided in section 1.6.2 of this thesis. To fulfill its purpose, the method proceeds by exploring and processing the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$, as discussed at the beginning of section 1.7, until a complete set of representatives of $\mathbf{H}$-congruence classes of chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is produced. The finiteness of the number of steps to be carried out to reach an end to the overall procedure is guaranteed by the fact that whenever X is a complex $K3$ surface, as indicated by Shimada in [19], the number of $\mathbf{H}$-congruence classes of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is finite. In order for Borcherds' method to be initiated, it must be provided with an initial chamber D_0 contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. From this chamber, the method starts its exploration of the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$. As stated in the section 4 of Shimada's article [19], classical theory provides a Weyl vector w_0 associated with a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ that may induce a suitable initial $\mathcal{P}_S$-chamber $D_0 = \mathcal{D}_0 \cap \mathcal{P}_S$ contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. When the method is provided with a starting point located within $\mathrm{Nef}(X) \cap \mathcal{P}_S$, we can then be sure that it will never leave the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$. Indeed, as discussed at the beginning of section 1.7, we know that the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is delimited by (-2)-walls. A key rule that the method must obey is that such walls are not to be crossed. Indeed, doing so would make the method leave the $\mathrm{Nef}(X) \cap \mathcal{P}_S$ area of study. In order to stay within $\mathrm{Nef}(X) \cap \mathcal{P}_S$, Borcherds' method relies on the procedure **RatDetect,** described in section 1.7.1. The purpose of this procedure consists in detecting (-2)-walls, so that the method can know if a wall can be safely crossed or should instead be avoided.

Two requirements have to be fulfilled:

- **Requirement n°1:** In order to induce a $\mathcal{P}_S$-chamber, the $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ must be $\iota(S)$-non-degenerate.

Depending on the embedding $\iota : S \hookrightarrow \mathbb{L}$, such a condition may or may not be fulfilled. Shimada provides a non-degeneracy criterion in his article [19]:

Shimada's non-degeneracy criterion: Assume that S is primitively embedded into $\mathbb{L}$ by $\iota : S \hookrightarrow \mathbb{L}$ and let $a \in \mathcal{P}_S$. Let $\mathcal{D}$ be a $\mathcal{P}_{\mathbb{L}}$-chamber with Weyl vector w. If the inequalities $\langle \mathrm{pr}_S(\iota(a)), q\rangle_{S^\vee} > 0$ hold for every $q \in \mathrm{pr}_S(\Delta_w)$, then $\mathcal{D}$ is $\iota(S)$-nondegerate. Note that a_S is contained in the interior of $D = \mathcal{D} \cap \mathcal{P}_S$, whenever these inequalities are satisfied, so that D is then a $\mathcal{P}_S$-chamber.

- **Requirement n°2:** Assuming that $\mathcal{D}_0$ is $\iota(S)$-non-degenerate, the induced chamber
$$D_0 = \mathcal{D}_0 \cap \mathcal{P}_S$$
must be contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$.

Shimada's non-degeneracy criterion can be applied to $\mathcal{D}_0$ with an ample class $a_0 \in \mathcal{P}_S$ to determine whether this requirement is fulfilled. Due to the limited scope of Shimada's non-degeneracy criterion, which is not generalistic, Shimada enforces a straightforward solution: Given an embedding $\iota : S \hookrightarrow \mathbb{L}$, a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ and an ample class $a_0 \in \mathcal{P}_S$ such that the non-degeneracy criterion fails; the section 8.3 of Shimada's article [19] contains the outline of a procedure that may produce an updated embedding

$$\iota_{\mathrm{upd}} : S \hookrightarrow \mathbb{L}$$

under which the non-degeneracy criterion applied to $\mathcal{D}_0$ and a_0 results in success. We dwell on this matter in section 1.8. Chambers are prominent objects of paramount importance within Borcherds' method.

Then comes the necessity to introduce a convention that will enable us to turn chambers into tangible data that can be processed at the scale of an implementation of Borcherds' method. We associate a tuple

$$D = \big(w_D, \mathcal{A}_{\mathbf{H}}(D), \Omega(D), \overline{\Omega}(D)\big)$$

to each $\mathcal{P}_S$-chamber D explored by Borcherds' method. The elements contained in this tuple can be described as follows:

- w_D with the Weyl vector of D computed using the procedure **WeylAdj** from section 1.7.2.
- $\Omega(D)$ is the set of walls D, computed by applying the procedures **DeltaW** and **SetOfWalls** from sections 1.5.2 and 1.5.1, respectively.
- $\mathcal{A}_{\mathbf{H}}(D)$ is a generating set of $\mathrm{Aut}_{\mathbf{H}}(D)$, computed by **AutChamber** from section 1.7.3.
- $\overline{\Omega}(D)$ is the set of walls of D taken with respect to anti-backtracking. That is, assuming that D is of level k, this set is a copy of $\Omega(D)$ from which the walls leading to chambers of level $k-1$ have been removed.

More details about the notion of anti-backtracking are provided online. We now assume that the Néron-Severi group $S = \mathrm{NS}(X)$ of the complex $K3$ surface X under study has been primitively embedded into a suitable even hyperbolic lattice $\mathbb{L}$ and further assume that an initial $\mathcal{P}_S$-chamber D_0 with Weyl vector w_0 contained into $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is known. As indicated at the beginning of section 1.7, the chamber D_0 is used as a reference point in order to layer the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ into various *levels*. The notion of level has been introduced in definition 27, earlier in this section.

Before proceeding further, let us get this straight about the notations that will be used until the end of this section. We denote by:

- $\mathcal{L}_k$ the set of chamber of level k. For example, we have $\mathcal{L}_0 = \{D_0\}$.
- $\mathbb{D}^*$ the set of sets $\mathcal{L}_k$ of chambers of various levels explored during the execution of Borcherds' method. Initially, $\mathbb{D}^* = \{\mathcal{L}_0\} = \{\{D_0\}\}$.
- Γ an initially empty set into which the generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ detected during the execution of the method will be stored.
- $\mathcal{R}_{\text{at}}$ an initially empty set into which will be stored the classes $m \in \mathrm{NS}(X)$ associated with the (-2)-walls detected by the procedure **RatDetect** among the elements of the sets of walls of the chambers explored by Borcherds' method during its execution.

We can now explain the chain of events occuring during an execution of Borcherds' method. Initially, we have

$$\Gamma = \{\}\,,\ \mathcal{R}_{\text{rat}} = \{\}\,,\ \mathbb{D}^* = \{\mathcal{L}_0\} \quad \text{and} \quad \mathcal{L}_0 = \{D_0\}\,.$$

Initialization - Chamber of level 0**:** The method starts by processing the initial $\mathcal{P}_S$-chamber D_0 with Weyl vector w_0. This step consists in computing the data tuple

$$D_0 = \big(w_0, \mathcal{A}_{\mathbf{H}}(D_0), \Omega(D_0), \overline{\Omega}(D_0)\big)$$

associated with D_0. From the input data of w_0, Borcherds' method calls for the procedure **DeltaW** described in section 1.5.1 to compute the set Δ_{w_0}. The projection $\mathrm{pr}_S(\Delta_{w_0})$ of Δ_{w_0} onto $S^\vee$ is then fed into the procedure **SetOfWalls.** The latter outputs the set $\Omega(D_0)$ of walls of D_0. The data of $\Omega(D_0)$ is then used as input into the procedure **AutChamber** (section 1.7.3) which produces a generating set $\mathcal{A}_{\mathbf{H}}(D_0)$ of $\mathrm{Aut}_{\mathbf{H}}(D_0)$.

Chambers of level 1**:** During this iteration, Borcherds' method explores and processes chambers of level 1 adjacent to chambers of level 0 within $\mathrm{Nef}(X) \cap \mathcal{P}_S$. That is, it will explore and process chambers adjacent to D_0 along its non (-2)-walls. Create an initially empty set $\mathcal{L}_1 = \{\}$ into which will be stored

the chambers of level 1 representing new $\mathbf{H}$-congruence classes of chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$ discovered during this iteration. Note that this stage, the only known congruence class is the class represented by D_0. The method then uses the procedure **RatDetect** to identify the (-2)-walls among the elements of $\Omega(D_0)$. Borcherds' method stores the classes in S of such walls into the set $\mathcal{R}_{\mathrm{rat}}$. For each non (-2)-wall $(m)^{\perp}$ of D_0, the method computes the Weyl vector $w_{D'}$ of the chamber D' adjacent to D_0 along $(m)^{\perp}$ by calling for the procedure **WeylAdj** with m and w_0 as input data. The data of the Weyl vector $w_{D'}$ thus obtained enables the method to compute the set of walls $\Omega(D')$ of the chamber D'. This is done in two steps.

- First, vector $w_{D'}$ is fed into **DeltaW**, which outputs the set $\Delta_{w_{D'}}$.
- The projection $\mathrm{pr}_{S^\vee}(\Delta_{w_{D'}})$ is then used as input into **SetOfWalls**, which returns the set $\Omega(D')$ of walls of the chamber D'.

The set $\Omega(D')$ is used as input into the procedure **AutChamber**, which outputs a set $\mathcal{A}_{\mathbf{H}}(D')$ of generators of $\mathrm{Aut}_{\mathbf{H}}(D')$. These generators are stored into the set Γ. Borcherds' method then needs to determine whether D' represents a brand new $\mathbf{H}$-congruence class: Since at this stage of the execution the class represented by D_0 is the only congruence class inventoried by the method. The only congruence test to be carried out by Borcherds' method during the exploration and processing of chambers of level 1 therefore consists in testing D' against D_0 for $\mathbf{H}$-congruency. To this end, the respective sets of walls $\Omega(D_0)$ and $\Omega(D')$ of D_0 and D' are used as input data into the procedure **CongChecker**:

- If **CongChecker** determines that D_0 and D' are not $\mathbf{H}$-congruent, then the chamber D' represents a brand new congruence class of chambers, and its associated data tuple

$$D' = \left(w_{D'}, \mathcal{A}_{\mathbf{H}}(D'), \Omega(D'), \overline{\Omega}(D')\right)$$

  is stored into the set $\mathcal{L}_1$ of representatives of congruence classes of chambers of level 1.

- If the result of the procedure **CongChecker** is that D_0 and D' are $\mathbf{H}$-congruent, at least one element of $g \in \mathbf{H}$ establishing the congruence is provided and stored into the set Γ.

Borcherds' method executes this routine until all the chambers adjacent to D_0 along its other non (-2)-walls have been explored an processed. When this is the case, the set $\mathcal{L}_1$ in stored into $\mathbb{D}^*$, so that we have $\mathbb{D}^* = \{\mathcal{L}_0, \mathcal{L}_1\}$. The method then proceeds as follows:

- If $\mathcal{L}_1 = \emptyset$, Borcherds' method ends its execution and outputs all the data collected during its execution.

Shimada indeed states in his article [19] that, in this case, the set

$$\mathbb{D} = \bigcup \mathbb{D}^*$$

is a complete set of representatives of $\mathbf{H}$-congruence classes of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ while the set Γ is a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$.

- If $\mathcal{L}_1 \neq \emptyset$, the method proceeds to its second iteration: The exploration and processing of chambers of level 2 by adjacency to chambers in $\mathcal{L}_1$.

Fast forwarding, we now assume that Borcherds' method has completed its k-th iteration. That is, let us assume that a non-empty set of representatives $\mathcal{L}_j$ has been obtained for each integer $j \leq k$ so that $\mathbb{D}^* = \{\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_k\}$.
We describe the $(k+1)$-th iteration of Borcherds' method.

Chambers of level $k+1$: During this iteration, Borcherds' method will explore and process chambers of level $k+1$ adjacent to chambers in $\mathcal{L}_k$ along their respective non (-2)-walls. An empty set $\mathcal{L}_{k+1} = \{\,\}$ is created, and will be used to store the $\mathbf{H}$-representatives of new congruence classes of chambers discovered during this iteration. For each $D \in \mathcal{L}_k$, and each element $m \in \Omega(D)$, Borcherds' method proceeds as follows:

The procedure **RatDetect** is used to determine whether $(m)^{\perp}$ is a (-2)-wall. When this is the case, a class in S associated with $(m)^{\perp}$ is returned by **RatDetect** and stored into the set $\mathcal{R}_{\text{rat}}$. If **RatDetect** outputs that $(m)^{\perp}$ is not a (-2)-walls, Borcherds' method explores the chamber D' adjacent to D along $(m)^{\perp}$. That is, the Weyl vector w' of the chamber D' adjacent to D along $(m)^{\perp}$ is computed by **WeylAdj** into which is fed the data of m and w. The set of walls of the chamber D' is computed in two steps:

- The element $w_{D'}$ is used as input into **DeltaW**, which returns $\Delta_{w'}$.
- The projection $\text{pr}_{S^{\vee}}(\Delta_{w'})$ is then used as input into **SetOfWalls**, which returns the set $\Omega(D')$ of walls of the chamber D'.

The data of the set of walls $\Omega(D')$ of D' is then fed as input into the procedure **AutChamber** which produces a generating set $\mathcal{A}_{\mathbf{H}}(D')$ of the group $\text{Aut}_{\mathbf{H}}(D')$. The elements of $\mathcal{A}_{\mathbf{H}}(D')$ are then stored into the set Γ of generators of

$$\text{Aut}_{\mathbf{H}}(\text{Nef}(X) \cap \mathcal{P}_S)$$

which have been detected since the execution of Borcherds' method. The method then determines whether D' represents a new $\mathbf{H}$-congruence class. To this end, the method tests D' is for $\mathbf{H}$-congruency against each chamber in

$$\begin{aligned}\mathbb{D} &= \bigcup \mathbb{D}^* \\ &= \mathcal{L}_0 \cup \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_k \cup \mathcal{L}_{k+1}.\end{aligned}$$

For each chamber $D'' \in \mathbb{D}$, the data of the respective sets

$$\Omega(D') \qquad \text{and} \qquad \Omega(D'')$$

of walls of D' and D'' is used as input into the procedure **CongChecker**.

If D' is not $\mathbf{H}$-congruent to at least one chamber in $\mathbb{D}$ then the chamber D' represents a new congruence class of chambers, and its associated data tuple

$$D' = \big(w_{D'}, \mathcal{A}_{\mathbf{H}}(D'), \Omega(D'), \overline{\Omega}(D')\big)$$

is stored into the set $\mathcal{L}_{k+1}$ of level $k+1$ representatives of $\mathbf{H}$-congruence classes. Otherwise, for each chamber D'' to which D' is $\mathbf{H}$-congruent, the associated elements $g \in \mathbf{H}$ returned by **CongChecker** are stored into the set Γ.

When the exploration and processing of the surroundings of each chamber in $\mathcal{L}_k$ has been performed, the method stores the set $\mathcal{L}_{k+1}$ into $\mathbb{D}^*$ and proceeds as follows:

- If $\mathcal{L}_{k+1} = \emptyset$, Borcherds' method ends its execution and outputs all the data collected during its execution.

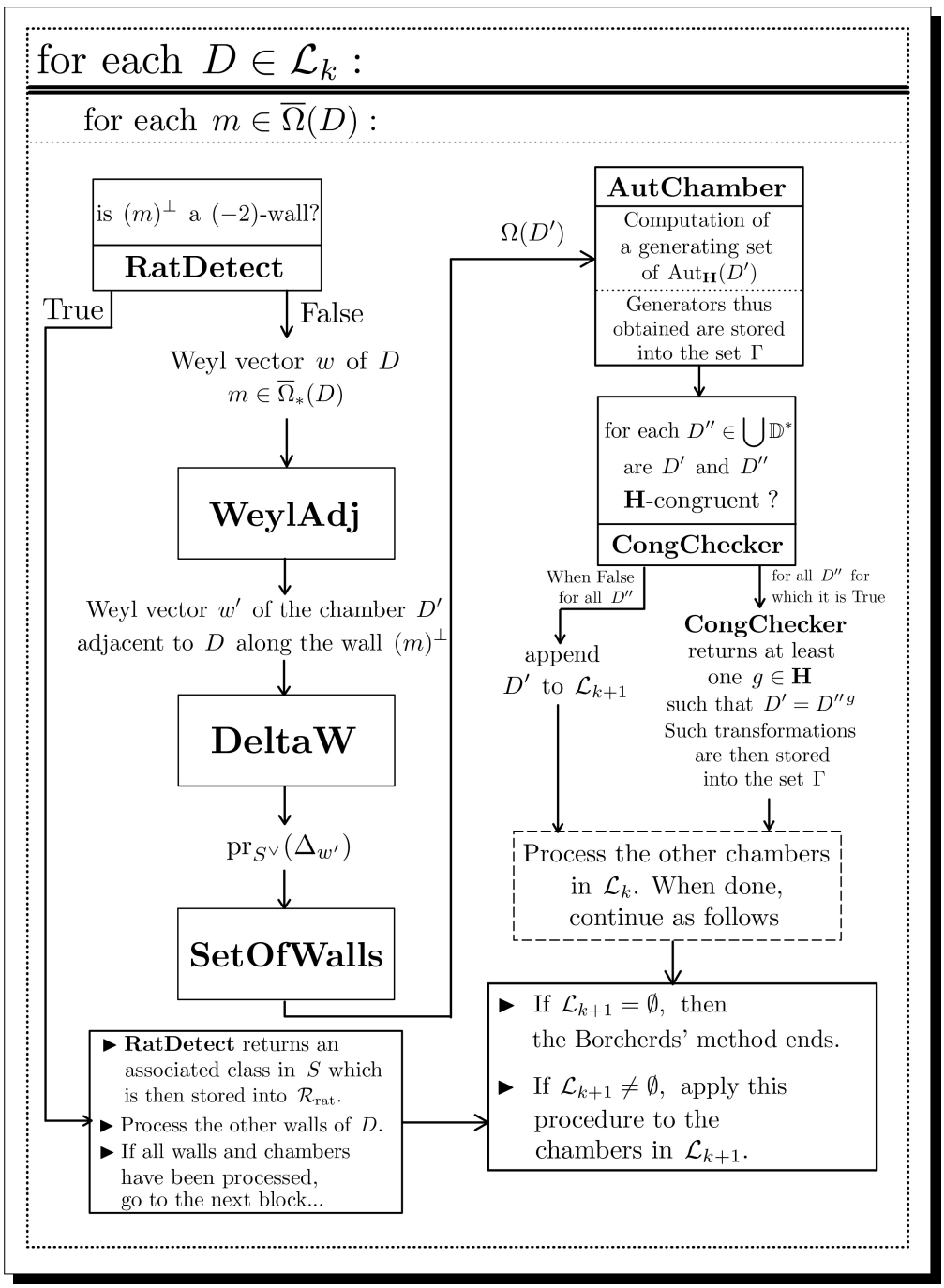
Indeed, by Shimada [19], the set

$$\mathbb{D} = \bigcup \mathbb{D}^*$$

is then a complete set of representatives of $\mathbf{H}$-congruence classes of chambers in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ while the set Γ is a generating set of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$.

- If $\mathcal{L}_{k+1} \neq \emptyset$, the method proceeds to its $(k+2)$-th iteration: The exploration and processing of chambers of chambers of level $k+2$ by adjacency to chambers contained in $\mathcal{L}_{k+1}$ along their respective non (-2)-walls.

Since theorem 3.7 from Shimada's article [19] ensures that the number of $\mathbf{H}$-congruence classes of chambers in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ is finite, the execution of Borcherds' method will end at one time or another, and will not run forever. The figure displayed on the following page illustrates the algorithmic structure of Borcherds' method.


for each $D \in \mathcal{L}_k$:
for each $m \in \overline{\Omega}(D)$:
is $(m)^\perp$ a (-2)-wall?
RatDetect
True
False
Weyl vector w of D
$m \in \overline{\Omega}_*(D)$
WeylAdj
Weyl vector w' of the chamber D'
adjacent to D along the wall $(m)^\perp$
DeltaW
$\mathrm{pr}_{S^\vee}(\Delta_{w'})$
SetOfWalls
$\Omega(D')$
AutChamber
Computation of a generating set of $\mathrm{Aut}_{\mathbf{H}}(D')$
Generators thus obtained are stored into the set Γ
for each $D'' \in \bigcup \mathbb{D}^*$ are D' and D'' $\mathbf{H}$-congruent ?
CongChecker
When False for all D''
for all D'' for which it is True
append D' to $\mathcal{L}_{k+1}$
CongChecker returns at least one $g \in \mathbf{H}$ such that $D' = D''^g$
Such transformations are then stored into the set Γ
Process the other chambers in $\mathcal{L}_k$. When done, continue as follows
▶ If $\mathcal{L}_{k+1} = \emptyset$, then the Borcherds' method ends.
▶ If $\mathcal{L}_{k+1} \neq \emptyset$, apply this procedure to the chambers in $\mathcal{L}_{k+1}$.
▶ RatDetect returns an associated class in S which is then stored into $\mathcal{R}_{\mathrm{rat}}$.
▶ Process the other walls of D.
▶ If all walls and chambers have been processed, go to the next block...

1.8 Embedding update procedure

We have seen in the previous section that executing Borcherds' method requires the data of a primitive embedding $\iota : S \hookrightarrow \mathbb{L}$ and of an initial $\mathcal{P}_{\mathbb{L}}$ -chamber $\mathcal{D}_0$ satisfying the two following conditions:

- The primitive embedding $\iota : S \hookrightarrow \mathbb{L}$ is such that the initial $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ is $\iota(S)$-nondegenerate, i.e., satisfies

  $$\mathrm{Int}(\mathcal{D}_0 \cap \mathcal{P}_S) \neq \emptyset$$

  so that it induces a $\mathcal{P}_S$-chamber

  $$D_0 = \mathcal{D}_0 \cap \mathcal{P}_S$$

  which can be used as a starting point for Borcherds' method to explore the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$.

- The induced $\mathcal{P}_S$-chamber D_0 must be contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$.

It turns out that it is enough to exhibit an ample class $a_0 \in \mathcal{P}_S$ such that

$$\iota(a_0) \in \mathrm{Int}(\mathcal{D}_0 \cap \mathcal{P}_S)$$

in order to ensure that the two above conditions are satisfied. Checking whether this condition holds can be done by using the procedure **Degentest** introduced in section 1.7: This procedure checks whether the strict inequality

$$\langle \mathrm{pr}_{S^\vee}(\iota(a_0)), q \rangle_{S^\vee} > 0 \tag{1.22}$$

holds for all elements $q \in \mathrm{pr}_{S^\vee}(\Delta_{w_0})$. When this is the case, the element $\iota(a_0)$ then belongs to the interior of $\mathcal{D}_0 \cap \mathcal{P}_S$ and the two above conditions are thus satisfied. The fact is that exhibiting an ample class $a_0 \in \mathcal{P}_S$ satisfying these conditions in the framework of a given embedding requires much luck. This issue must therefore be approached from another angle. Given an initial primitive

embedding

$$\iota : S \hookrightarrow \mathbb{L},$$

an ample class a_0 and the Weyl vector w_0 of the initial chamber $\mathcal{D}_0$, Shimada provides in the section 8 of his article [19] the outline of a procedure which may possibly yield a transformation

$$\tau : \mathbb{L} \longrightarrow \mathbb{L}$$

such that

$$\tau \circ \iota : S \hookrightarrow \mathbb{L}$$

is an updated primitive embedding under which the inequalities

$$\left\langle \mathrm{pr}_{S^\vee} \left((\tau \circ \iota)(a_0) \right), q \right\rangle_{S^\vee} > 0 \tag{1.23}$$

are satisfied for all elements $q \in \mathrm{pr}_{S^\vee}(\Delta_{w_0})$. That is,

$$(\tau \circ \iota)(a_0) \in \mathrm{Int}(\mathcal{D}_0 \cap \mathcal{P}_S).$$

Note that τ is obtained as the composition of various reflections with respect to carefully chosen elements of $\mathcal{R}_{\mathbb{L}}$. The updated embedding

$$\iota_{\mathrm{upd}} = \tau \circ \iota$$

thus obtained then provides a framework under which the $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ is $\iota_{\mathrm{upd}}(S)$-nondegenerate and thus induces a $\mathcal{P}_S$-chamber $D_0 = \mathcal{D}_0 \cap \mathcal{P}_S$ contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. From our point of view, Shimada's approach is the best possible course of action to deal with the issue of finding a non-degenerate chamber by acting at directly on the embedding of S into $\mathbb{L}$. The issue is that this approach may require many attemps, a lot of trials and failures, before eventually resulting in a positive outcome. We took care of this issue.
In this section, we proceed along three axes:

- Assuming given an embedding

$$\iota : S \hookrightarrow \mathbb{L},$$

  an ample class $a_0 \in \mathcal{P}_S$ and a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$, we will start by investigating the causes the failure of the non-degeneracy condition for $\mathcal{D}_0$.

- We will then present Shimada's original idea to update an embedding.

- We will finally explain how we modernized and improved Shimada's idea.

1.8.1 Failure of the non-degeneracy condition, a quick survey

Assume that an initial embedding

$$\iota : S \hookrightarrow \mathbb{L},$$

a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ and an ample class $a_0 \in \mathcal{P}_S$ such that

$$\iota(a_0) \notin \mathrm{Int}(\mathcal{D}_0 \cap \mathcal{P}_S) \tag{1.24}$$

are given. Using the elementary fact that

$$\mathrm{Int}(\mathcal{D}_0 \cap \mathcal{P}_S) = \mathrm{Int}(\mathcal{D}_0) \cap \mathrm{Int}(\mathcal{P}_S)$$

we see that the condition (1.24) holds if and only if either

$$\iota(a_0) \notin \mathrm{Int}(\mathcal{D}_0) \quad \text{or} \quad \iota(a_0) \notin \mathrm{Int}(\mathcal{P}_S)$$

Since a_0 is ample and thus belongs to $\mathcal{P}_S$ by assumption, we see that

$$\iota(a_0) \notin \mathrm{Int}(\mathcal{D}_0)$$

Two possibilities should then be considered

- The element $\iota(a_0)$ belongs to the boundary of $\mathcal{D}_0$, i.e.,

$$\iota(a_0) \in \mathcal{D}_0 \backslash \operatorname{Int}(\mathcal{D}_0)$$

  where we recall that chambers are by definition closed sets. In this case, there exists at least one $q \in \Omega(\mathcal{D}_0) \subset \mathcal{R}_{\mathbb{L}}$ such that $\langle q, \iota(a_0)\rangle_{\mathbb{L}} = 0$.

In order words, the element $\iota(a_0)$ is *stuck* in a wall of $\mathcal{D}_0$. Here we touch on a point which is key to understand why Shimada's idea may fail. In case $\iota(a_0)$ belongs to a wall of $\mathcal{D}_0$, application of reflections which respect to elements of $\mathcal{R}_{\mathbb{L}}$ will not move $\iota(a_0)$ by a single inch since walls of $\mathcal{P}_{\mathbb{L}}$-chambers are themselves elements of $\mathcal{R}_{\mathbb{L}}$ and are by definition left invariant by such transformations. Keep this fact in mind, it will be useful during the next section. The other possibility to be considered is simple:

- The element $\iota(a_0)$ does not belong to $\mathcal{D}_0$. That is, there exists at least one element $q \in \Omega(\mathcal{D}_0)$ such that $\langle q, \iota(a_0))\rangle_{\mathbb{L}} < 0$.

In this case, applying reflections with respect to Weyl chosen elements of $\mathcal{R}_{\mathbb{L}}$ may succeed in order to obtain an updated embedding ι_{upd} such that

$$\iota_{\text{upd}}(a_0) \in \mathcal{D}_0 \cap \mathcal{P}_S.$$

1.8.2 Shimada's embedding update procedure

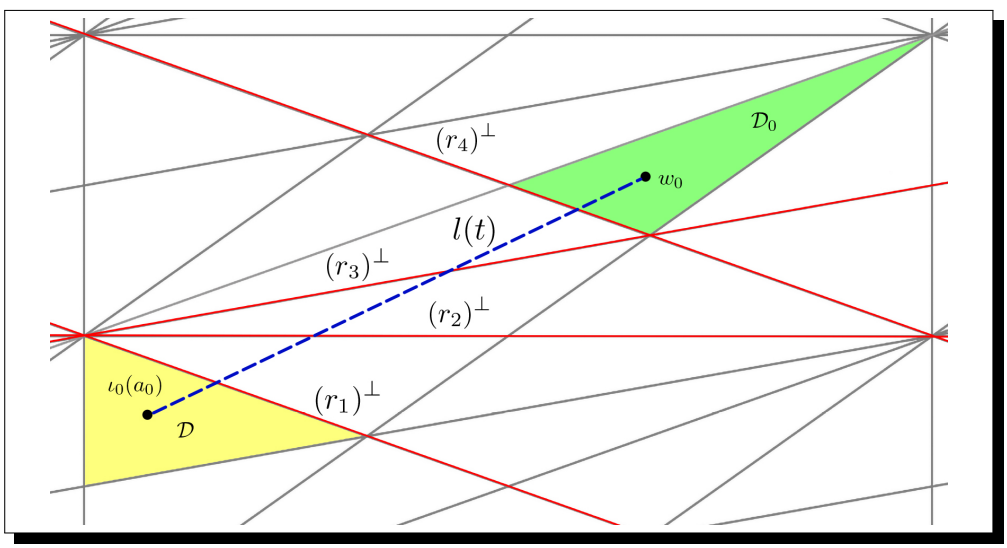
Assume that $\iota(a_0) \notin \mathcal{D}_0$, so that $\iota(a_0)$ belongs to another $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D} \neq \mathcal{D}_0$. In order to avoid the above-mentioned issue where $\iota(a_0)$ would be *stuck* in a wall, assume furthermore that $\iota(a_0) \in \operatorname{Int}(\mathcal{D})$. Denote by w_0 the Weyl vector of $\mathcal{D}_0$ and recall that the definition of a Weyl vector implies that the Weyl vector of a $\mathcal{P}_{\mathbb{L}}$-chamber is contained in its interior. We therefore have $w_0 \in \operatorname{Int}(\mathcal{D}_0)$. Let

$$l(t) = (1-t)\iota(a_0) + tw_0, \qquad 0 \leq t \leq 1$$

be the line segment in $\mathcal{P}_{\mathbb{L}}$ connecting $\iota(a_0)$ to w_0. Since $\iota(a_0)$ and w_0 do not belong to the same chamber, this segment must intersect some walls $(r_i)^{\perp}$ induced by elements $r_i \in \mathcal{R}_{\mathbb{L}}$. In order to have a clear view of what happens, assume that $l(t)$ intersects the walls

$$(r_1)^{\perp}, (r_2)^{\perp}, (r_3)^{\perp}, (r_4)^{\perp}$$

induced $r_1, r_2, r_3, r_4 \in \mathcal{R}_{\mathbb{L}}$ so that the situation is illustrated as follows:



We can see that $\iota(a_0)$ is located in the interior of a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}$ colored in yellow. We can also see that that w_0 is located in the interior of a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ colored in green. The path $l(t)$ connecting $\iota(a_0)$ to w_0 in $\mathcal{P}_{\mathbb{L}}$ is represented as a dashed line, colored in dark blue. Moreover, $l(t)$ is here represented as crossing four walls $(r_i)^{\perp}$ with $r_i \in \mathcal{R}_{\mathbb{L}}$ for $i = 1, 2, 3, 4$, which have been highlighted as red lines. Note that the locations displayed on this figure imply that

$$\langle w_0, r_i \rangle > 0$$

while

$$\langle \iota(a_0), r_i \rangle < 0$$

for $i = 1, 2, 3, 4$. Recall that to each element $r \in \mathcal{R}_{\mathbb{L}}$ can be associated a reflection

$$s_r : \mathbb{L} \longrightarrow \mathbb{L}$$

$$s_r : x \longmapsto x + \langle x, r \rangle_{\mathbb{L}} \, r.$$

with respect to the hyperplane $(r)^{\perp}$. Shimada's idea consists in successively applying the reflections s_{r_i} for $i \in \{1, 2, 3, 4\}$ to the embedding

$$\iota : S \hookrightarrow \mathbb{L}$$

so that an updated embedding

$$\tau \circ \iota : S \hookrightarrow \mathbb{L} \qquad \text{with} \qquad \tau = s_{r_4} \circ s_{r_3} \circ s_{r_2} \circ s_{r_1}$$

is obtained and hopefully provides a framework under which the condition

$$(\tau \circ \iota)\,(a_0) \in \text{Int}(\mathcal{D}_0 \cap \mathcal{P}_S)$$

is satisfied. More generally, given an ample class a_0, a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ with Weyl vector w_0 and an initial embedding $\iota : S \hookrightarrow \mathbb{L}$ such that

$$\iota(a_0) \notin \textbf{Int}(\mathcal{D}_0 \cap \mathcal{P}_S),$$

Shimada enforces an approach which consists in proceeding as follows:

Step n°1 - Using Shimada's algorithm 3.3 from his article [18], compute the set

$$\mathcal{M} = \{r \in \mathcal{R}_{\mathbb{L}} \mid \langle \iota(a_0), r \rangle_{\mathbb{L}} < 0, \, \langle w_0, r \rangle_{\mathbb{L}} > 0\}$$

of elements $r \in \mathcal{R}_{\mathbb{L}}$ such that $(r)^{\perp}$ seperates $\iota(a_0)$ from w_0. We implemented this algorithm and named it **ShiBooster**. More details about the latter are available in the second part of this thesis and on K3surfaces.com.

Step n°2 - Shimada then defines the line segment

$$l(t) = (1-t)\iota(a_0) + tw_0, \qquad 0 \leq t \leq 1$$

connecting $\iota(a_0)$ to w_0 in $\mathcal{P}_{\mathbb{L}}$. For each element $r \in \mathcal{M}$, we have to solve

$$\langle l(t), r \rangle_{\mathbb{L}} = 0$$

for $0 \leq t \leq 1$.

Step n°3 - Elements of $\mathcal{M}$ are then re-labelled in such a way that elements $r_i, r_j \in \mathcal{M}$ with respective associated solutions t_i, t_j satisfy

$$i < j \quad \text{if and only if} \quad t_i < t_j.$$

Note that Shimada requires that all the t_i should be distinct, and orders to pick another ample class and try again until this requirement is fulfilled. Assume that the elements of $\mathcal{M}$ have thus been relabelled in such a way that the set $\mathcal{M}$ can be expressed as

$$\mathcal{M} = \{r_1, r_2, \ldots, r_N\}.$$

The segment $l(t)$ then intersects the walls $(r_i)^{\perp}$ according to the order which arises from the labelling of the elements $r_i \in \mathcal{M}$.

Step n°4: Define

$$\tau = s_{r_N} \circ s_{r_{N-1}} \circ \cdots \circ s_{r_2} \circ s_{r_1}$$

and update the initial embedding as

$$\tau \circ \iota : S \hookrightarrow \mathbb{L}.$$

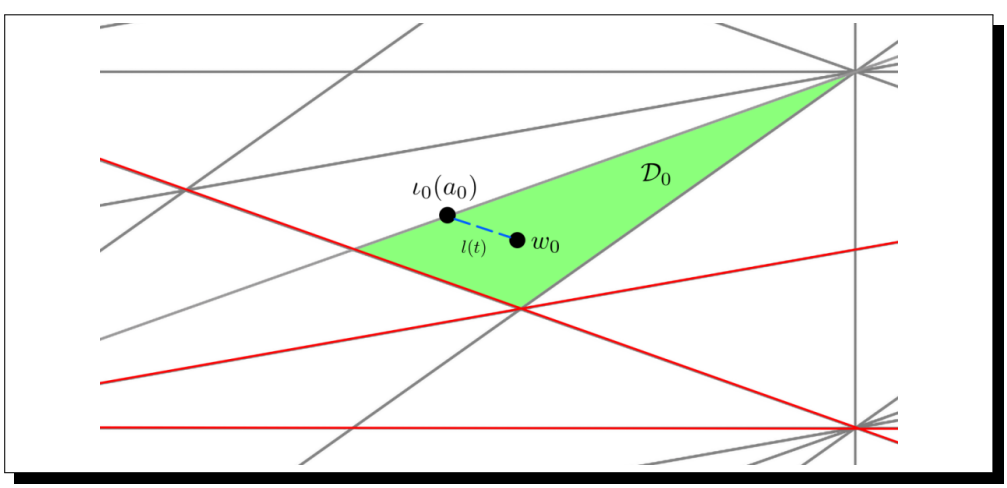
Step n°5: Use Shimada's non-degeneracy criterion (procedure **DegenTest** from section 1.7) to $\mathcal{D}_0$ and $(\tau \circ \iota)(a_0)$ in order to check whether

$$(\tau \circ \iota)(a_0) \in \mathrm{Int}(\mathcal{D}_0 \cap \mathcal{P}_S)$$

holds under the framework of the updated embedding. We explain in the following section how Shimada's embedding update procedure can be improved. His procedure suffers from the fact that many attempts with various ample classes a_0 may be required before eventually obtaining a positive result. Also, the procedure may not work at all, and no explanation regarding this fact is provided in Shimada's article [19]. If we remember our discussion from section 1.8.1, we see that Shimada's procedure will fail whenever $\iota(a_0) \in (r)^{\perp}$ for some $r \in \mathcal{R}_{\mathbb{L}}$. In such cases, we say that $\iota(a_0)$ is stuck into a wall. It is therefore important to make sure that the set

$$\{r \in \mathcal{R}_{\mathbb{L}} \mid \langle \iota(a_0), r \rangle_{\mathbb{L}} = 0\}$$

is empty before enforcing Shimada's embedding update procedure with. The situation is otherwise especially problematic when $\iota(a_0)$ is stuck into a wall of the initial $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$, as illustrated in the figure below.



When such a situation occurs, there is no leeway: The mechanics of the embedding update procedure rely on the application of reflections with respect to walls crossing the path between $\iota(a_0)$ and w_0. When $\iota(a_0)$ is stuck into a wall of $\mathcal{D}_0$, there is no wall separating it from w_0, except the wall into which it is stuck. Since reflections of the form

$$s_r : x \longmapsto x + \langle x, r \rangle_{\mathbb{L}} \, r$$

act as the identity on elements $x \in (r)^{\perp}$, we cannot do anything to free $\iota(a_0)$ from the wall into which it is stuck. In fact, the only thing that can be done consists in either finding another primitive embedding or finding another ample class. It should be noted that a decade ago, Shimada provided no explanation on why he asked his readers to use another ample class when the procedure outlined in his 2013 article fails. We hope that our explanations provide a better understanding of what is happening behind the scenes.

1.8.3 A new perspective on Shimada's embedding update procedure

We now explain how we improved Shimada's embedding update procedure. As before, we assume given an ample class $a_0 \in \mathcal{P}_S$, a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ and an initial primitive embedding $\iota : S \hookrightarrow \mathbb{L}$ such that $\iota(a_0) \notin \mathcal{D}_0$ but is instead contained in the interior of a $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D} \neq \mathcal{D}_0$. We also recall that the Weyl vector w_0 of the $\mathcal{P}_{\mathbb{L}}$-chamber $\mathcal{D}_0$ satisfies $w_0 \in \mathrm{Int}(\mathcal{D}_0)$ and compute the set. We let

$$l(t) = (1-t)\iota(a_0) + t w_0 \, , \qquad 0 \leq t \leq 1$$

be the line segment connecting $\iota(a_0)$ to w_0 in $\mathcal{P}_{\mathbb{L}}$. Using our implementation **ShiBooster** of Shimada's algorithm 3.3 from [18], we compute the set

$$\mathcal{M} = \{r \in \mathcal{R}_{\mathbb{L}} \mid \langle \iota(a_0), r \rangle_{\mathbb{L}} < 0,\ \langle w_0, r \rangle_{\mathbb{L}} > 0\}$$

of elements $r \in \mathcal{R}_{\mathbb{L}}$ such that $(r)^{\perp}$ separates $\iota(a_0)$ from w_0 and is thus crossed by the path defined by $l(t)$. We assume that

$$\mathcal{M} = \{r_1, r_2, \ldots, r_N\}$$

and that the elements in $\mathcal{M}$ are labelled in such a way that $r_i, r_j \in \mathcal{M}$ satisfy

$$i < j \quad \text{if and only if} \quad t_i < t_j$$

where t_k is the solution of

$$\langle l(t_k), r_k \rangle_{\mathbb{L}} = 0$$

The walls induced by elements of $\mathcal{M}$ should be considered **obstructions** on the path defined by the line segment $l(t)$: As t increases from 0 to 1, the path successively encounters the wall $(r_1)^{\perp}$, then $(r_2)^{\perp}$, ..., and finally $(r_N)^{\perp}$. Applying a reflection

$$s_{r_i} : x \longmapsto x + \langle x, r_i \rangle_{\mathbb{L}}\, r_i$$

to the embedding $\iota : S \hookrightarrow \mathbb{L}$ amounts to sending $\iota(a_0)$ to the other side of the wall $(r_i)^{\perp}$, hence clearing the obstruction represented by this wall. For example, assume that the reflection s_{r_1} with respect to r_1 is applied to the embedding ι, so that we have an updated embedding

$$s_{r_1} \circ \iota : S \hookrightarrow \mathbb{L}$$

The inequality satisfied

$$\langle \iota(a_0), r_1 \rangle_{\mathbb{L}} < 0$$

in the framework of the initial embedding can then be turned into

$$\langle (s_{r_1} \circ \iota)(a_0), r_1 \rangle_{\mathbb{L}} > 0$$

in the framework of the updated embedding $s_{r_1} \circ \iota$. It is clear that the wall $(r_1)^{\perp}$ associated with $r_1 \in \mathcal{M}$ is therefore no longer an obstruction. The fundamental difference between our approach and Shimada's is that we consider the following question: In the framework of the updated embedding $s_{r_1} \circ \iota$, are the walls associated with elements of $\mathcal{M} \setminus \{r_1\}$ still obstructions? For example, we can legitimately wonder whether $(r_2)^{\perp}$ an is still an obstruction separating $(s_{r_1} \circ \iota)(a_0)$ from w_0. That is, do we need do apply the reflection s_{r_2} to $s_{r_1} \circ \iota$? Another perfectly legitimate consideration consists in wondering whether the application of s_{r_1} did introduce new obstructions in the framework of the updated embedding? The only way to have answers consists in computing the set $\mathcal{M}$ of obstructions again, this time taking into account the fact that embedding has been updated. The result is an iterative procedure: We start by clearing the obstruction closest to $\iota(a_0)$ by applying s_{r_1}. We then compute the updated set of obstructions and clear the obstruction closest to $s_{r_1} \circ \iota(a_0)$, and continue. When the updated set of obstructions is the empty set, the procedure terminates.

Iteration n°1 - We compute the initial set of obstructions

$$\mathcal{M}_1 = \{r \in \mathcal{R}_{\mathbb{L}} \mid \langle \iota(a_0), r \rangle_{\mathbb{L}} < 0,\ \langle w_0, r \rangle_{\mathbb{L}} > 0\}$$

For each element $r \in \mathcal{M}_1$, we then solve for t the equation

$$\langle l(t), r \rangle_{\mathbb{L}} = 0.$$

We drop Shimada's requirement that no two elements of $\mathcal{M}$ should have the same associated solution. We determine the smallest associated solution and randomly pick an element say associated with this solution, say $r_1 \in \mathcal{M}_1$. We

then apply the reflection

$$s_{r_1} : x \longmapsto x + \langle x, r_1 \rangle_{\mathbb{L}} \, r_1$$

to the embedding $\iota : S \hookrightarrow \mathbb{L}$ so that we have an updated embedding

$$s_{r_1} \circ \iota : S \hookrightarrow \mathbb{L}.$$

which provides a framework under which $(r_1)^{\perp}$ is not an obstruction separating $(s_{r_1} \circ \iota)(a_0)$ from w_0.

Iteration n°2 - We now compute the set of obstructions in order to take into account the fact that the embedding has been updated and that obstructions separating $(s_{r_1} \circ \iota)(a_0)$ from w_0 may not be the same obstructions than those which separated $\iota(a_0)$ from w_0. We thus compute

$$\mathcal{M}_2 = \{r \in \mathcal{R}_{\mathbb{L}} \mid \langle (s_{r_1} \circ \iota)(a_0), r \rangle_{\mathbb{L}} < 0, \ \langle w_0, r \rangle_{\mathbb{L}} > 0\}.$$

If $\mathcal{M}_2 = \emptyset$, the procedure stops and the updated embedding is

$$\iota_{\text{upd}} = s_{r_1} \circ \iota.$$

Otherwise, we solve for t the equation

$$\langle l(t), r \rangle_{\mathbb{L}} = 0$$

for each element $r \in \mathcal{M}_2$. We then pick one of the elements, say $r_2 \in \mathcal{M}_2$, associated with the smallest solution. We then apply reflection s_{r_2} to the embedding $s_{r_1} \circ \iota$ so that we have an updated embedding

$$s_{r_2} \circ s_{r_1} \circ \iota : S \hookrightarrow \mathbb{L}$$

under which the wall $(r_2)^{\perp}$ is not an obstruction for $(s_{r_2} \circ s_{r_1} \circ \iota)(a_0)$, that is,

such that

$$\langle (s_{r_2} \circ s_{r_1} \circ \iota)(a_0), r_2 \rangle_{\mathbb{L}} > 0$$

holds.

Fast forwarding, we assume that k-th iteration of the procedure has been accomplished so that

$$\tau_k \circ \iota : S \hookrightarrow \mathbb{L} \qquad \text{with} \qquad \tau_k = s_{r_k} \circ s_{r_{k-1}} \circ \cdots \circ s_{r_2} \circ s_{r_1}$$

has been obtained.

$(k+1)$**-th iteration** - In order to compute an updated list of obstructions, we compute

$$\mathcal{M}_{k+1} = \{ r \in \mathcal{R}_{\mathbb{L}} \mid \langle (\tau_k \circ \iota)(a_0), r \rangle_{\mathbb{L}} < 0,\ \langle w_0, r \rangle_{\mathbb{L}} > 0 \}.$$

If $\mathcal{M}_{k+1} = \emptyset$, the procedure terminates and we use

$$\iota_{\text{upd}} = \tau_k \circ \iota.$$

as our updated embedding. Otherwise, we solve for t the equation

$$\langle l(t), r \rangle_{\mathbb{L}} = 0$$

for each element $r \in \mathcal{M}_{k+1}$. We pick an element, say $r_{k+1} \in \mathcal{M}_{k+1}$ associated with the smallest solution obtained and apply the reflection $s_{r_{k+1}}$ to $\tau_k \circ \iota$. That is, we define an updated embedding

$$\tau_{k+1} \circ \iota : S \hookrightarrow \mathbb{L}$$

with

$$\tau_{k+1} = s_{r_{k+1}} \circ \tau_k.$$

1.9 Fundamental domain, associated cone, Hilbert Basis

Let G be a group and Y a set on which G acts on the left. We denote by gx the image of an element $x \in Y$ by an element $g \in G$. Given a subset $F \subset Y$, we denote by

$$g(F) = \{gx \mid x \in F\}$$

the image of F under the action of $g \in G$. We recall that a fundamental domain for the action of a group G on a set Y is a subset $F \subset Y$ having the following properties:

- $\bigcup_{g \in G} g(\overline{F}) = Y$
- The intersection $g(F) \cap h(F)$ is empty for all $g, h \in G$ such that $g \neq h$.

Assume that Borcherds' method has been executed and returned a set

$$\mathbb{D}^* = \{\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_m\}$$

where $\mathcal{L}_j$ denotes the set of representatives of level j of $\mathbf{H}$-congruence classes of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. As indicated in Shimada's article [19] section 6, the union

$$\begin{aligned} \mathbb{D} &= \bigcup_{\mathcal{L} \in \mathbb{D}^*} \mathcal{L} \\ &= \mathcal{L}_0 \cup \mathcal{L}_1 \cup \cdots \cup \mathcal{L}_m \end{aligned}$$

is a complete set of representatives of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$-congruence classes of chambers, i.e., of $\mathbf{H}$-congruence classes of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. The set $\mathbb{D}$ thus contains exactly one representative of each $\mathbf{H}$-congruence class of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. Let $D \in \mathbb{D}$. We denote by

$$\mathcal{F}(D) \subset D$$

a fundamental domain of the action of $\mathrm{Aut}_{\mathbf{H}}(D) \subset \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ on D.

We now establish the following important proposition:

Proposition 30. *The union* $\bigcup_{D\in\mathbb{D}} \mathcal{F}(D)$ *is a fundamental domain of the action of* $Aut_{\mathbf{H}}(Nef(X)\cap\mathcal{P}_S)$ *on* $Nef(X)\cap\mathcal{P}_S$.

Proof. In order to simplify the notations, we will make use of the shorthands

$$\mathcal{N}_X = \mathrm{Nef}(X)\cap\mathcal{P}_S$$

and

$$\mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X) = \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X)\cap\mathcal{P}_S)$$

when necessary. Define

$$\mathscr{F} = \bigcup_{D\in\mathbb{D}} \mathcal{F}(D).$$

Let us apply the definition of a fundamental domain stated at the beginning of this section. We establish the two following properties

- There is an equality

$$\bigcup_{g\in\mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathscr{F}}) = \mathrm{Nef}(X)\cap\mathcal{P}_S.$$

- The implication

$$g\neq h \Longrightarrow g(\mathscr{F})\cap h(\mathscr{F})\neq\emptyset$$

  holds for $g,h\in\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X)\cap\mathcal{P}_S)$.

Let us begin by establishing the inclusion

$$\mathrm{Nef}(X)\cap\mathcal{P}_S \quad\subseteq \bigcup_{g\in\mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathscr{F}}).$$

Let $p_1\in\mathrm{Nef}(X)\cap\mathcal{P}_S$. We have seen at the beginning of section 1.7 that $\mathrm{Nef}(X)\cap\mathcal{P}_S$ is tiled by $\mathcal{P}_S$-chambers. Consequently, we have $p_1\in D$ for some

$\mathcal{P}_S$-chamber $D \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$. Assume that

$$\mathbb{D} = \{D_0, D_1, \ldots, D_r\}$$

for some integer $r \geq 0$. Since $\mathbb{D}$ is a complete set of representatives of $\mathbf{H}$-congruence classes of $\mathcal{P}_S$-chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$, the class of the chamber D possesses a representative $D_k \in \mathbb{D}$. Moreover, there exists an element

$$g_1 \in \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$$

sending D onto the representative of its congruence class, that is, such that

$$D^{g_1} = D_k.$$

The transformation g_1 hence sends $p_1 \in D$ to an element $p_1^{g_1} \in D_k$. Let

$$\mathcal{F}(D_k) \subset D_k$$

be the fundamental of the action of $\mathrm{Aut}_{\mathbf{H}}(D_k)$ on D_k. By definition of a fundamental domain, there exists an element

$$p_2 \in \mathcal{F}(D_k)$$

and a group element

$$g_2 \in \mathrm{Aut}_{\mathbf{H}}(D_k)$$

such that

$$p_2 = p_1^{g_1 g_2}.$$

Hence, we have

$$p_1 = p_2^{g_2^{-1} g_1^{-1}}$$

with

$$g_2^{-1} g_1^{-1} \in \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S).$$

We have

$$p_2 \in \mathcal{F}(D_k) \subseteq \mathscr{F} \subseteq \overline{\mathscr{F}}$$

and thus

$$p_2^{g_2^{-1} g_1^{-1}} \in \bigcup_{g \in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathscr{F}}).$$

We thus established that

$$p_1 \in \mathrm{Nef}(X) \cap \mathcal{P}_S \Longrightarrow p_1 \in \bigcup_{g \in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathscr{F}}),$$

that is,

$$\mathrm{Nef}(X) \cap \mathcal{P}_S \subseteq \bigcup_{g \in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathscr{F}}).$$

We now establish the reverse inclusion. Let D be a $\mathcal{P}_S$-chamber over $\mathrm{Nef}(X) \cap \mathcal{P}_S$, that is,

$$D \subset \mathrm{Nef}(X) \cap \mathcal{P}_S.$$

Let $\mathcal{F}(D) \subseteq D$ be a fundamental domain for the action of $\mathrm{Aut}_{\mathbf{H}}(D)$ on D. By definition of a fundamental domain, the equality

$$\bigcup_{g \in \mathrm{Aut}_{\mathbf{H}}(D)} g(\overline{\mathcal{F}(D)}) = D$$

holds. Combining this equality to the fact that

$$\mathrm{Aut}_{\mathbf{H}}(D) \subset \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$$

yields

$$\begin{aligned} \bigcup_{g \in \mathrm{Aut}_{\mathbf{H}}(D)} g(\overline{\mathcal{F}(D)}) &\subseteq \bigcup_{g \in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathcal{F}(D)}) \\ &\subseteq \mathrm{Nef}(X) \cap \mathcal{P}_S \end{aligned}$$

Since this inclusion holds for any $\mathcal{P}_S$-chamber D contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$ and

thus holds for any chamber $D \in \mathbb{D}$, we have

$$\bigcup_{D\in\mathbb{D}} \left(\bigcup_{g\in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathcal{F}(D)}) \right) \subset \mathrm{Nef}(X) \cap \mathcal{P}_S. \tag{1.25}$$

In the section 6 of [19], Shimada establishes that $\mathbb{D}$ is a finite set. Using the fact that the closure of a finite union of sets is equal to the union of closures, we have

$$\overline{\mathscr{F}} = \overline{\bigcup_{D\in\mathbb{D}} \mathcal{F}(D)} = \bigcup_{D\in\mathbb{D}} \overline{\mathcal{F}(D)}.$$

Hence, there is an equality

$$\begin{aligned} \bigcup_{g\in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g(\overline{\mathscr{F}}) &= \bigcup_{g\in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} g\left(\overline{\bigcup_{D\in\mathbb{D}} \mathcal{F}(D)} \right) \\ &= \bigcup_{g\in \mathrm{Aut}_{\mathbf{H}}(\mathcal{N}_X)} \bigcup_{D\in\mathbb{D}} g\overline{\mathcal{F}(D)}. \end{aligned}$$

Combining this equality to the inclusion of expression (1.25) leads us to

$$\bigcup_{g\in \mathrm{Aut}_{\mathbf{H}}(N_X)} g(\overline{\mathscr{F}}) \subseteq \mathrm{Nef}(X) \cap \mathcal{P}_S.$$

Since we established ealier the opposite direction of this inclusion, we deduce that

$$\bigcup_{g\in \mathrm{Aut}_{\mathbf{H}}(N_X)} g(\overline{\mathscr{F}}) = \mathrm{Nef}(X) \cap \mathcal{P}_S,$$

as desired. We will use a proof by contradiction to establish that

$$g(\mathscr{F}) \cap h(\mathscr{F}) = \emptyset$$

holds for any two distinct elements $g, h \in \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$. Assume that

there exist elements $g, h \in \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$, with $g \neq h$ such that

$$g(\mathscr{F}) \cap h(\mathscr{F}) \neq \emptyset.$$

Let $p \in g(\mathscr{F}) \cap h(\mathscr{F})$. Then there exist $\mathcal{P}_S$-chambers $D, D' \in \mathbb{D}$ such that

$$p \in g(\mathcal{F}(D)) \subseteq g(D) \qquad \text{and} \qquad p \in h(\mathcal{F}(D')) \subset h(D').$$

That is,

$$g(D) \cap h(D') \neq 0.$$

Recall that $D, D' \in \mathbb{D}$ and that $\mathbb{D}$ is assumed to be a complete set of representatives of $\mathbf{H}$-congruence classes of chambers contained in $\mathrm{Nef}(X) \cap \mathcal{P}_S$. Hence

$$g(\mathscr{F}) \cap h(\mathscr{F}) = \emptyset.$$

The union

$$\mathscr{F} = \bigcup_{D \in \mathbb{D}} \mathcal{F}(D)$$

is therefore a fundamental domain of the action of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ onto $\mathrm{Nef}(X) \cap \mathcal{P}_S$, as desired. □

Assume that $\mathrm{Aut}_{\mathbf{H}}(D) = \{\mathrm{Id}\}$ holds for all elements of $\mathbb{D}$. In this case, the equality

$$\mathcal{F}(D) = D$$

holds for all $D \in \mathbb{D}$. An immediate consequence of proposition 30 is then that the union

$$\bigcup_{D \in \mathbb{D}} D$$

is a fundamental domain of the action of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ on $\mathrm{Nef}(X) \cap \mathcal{P}_S$. Recall that whenever the $K3$ surface X under study satisfies

$$\rho_X < 20 \qquad \text{and} \qquad -1 \notin \mathrm{Ker}(\eta_T)$$

theorem 22 states that there is an isomorphism

$$\mathrm{Aut}(X) \simeq \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S).$$

In this case, the assumption that $\mathrm{Aut}_{\mathbf{H}}(D) = \{\mathrm{Id}\}$ holds for all $D \in \mathbb{D}$ combined with proposition 30 implies that

$$\bigcup_{D \in \mathbb{D}} D$$

is a fundamental domain of the action of $\mathrm{Aut}(X)$ on $\mathrm{Nef}(X) \cap \mathcal{P}_S$, that is

Corollary 31. *Assume that X satisfies the conditions of theorem 22 and that $Aut_{\mathbf{H}}(D) = \{Id\}$ holds for all $D \in \mathbb{D}$. Then the union $\bigcup_{D \in \mathbb{D}} D$ is a fundamental domain of the action of $Aut(X)$ on $Nef(X) \cap \mathcal{P}_S$.*

1.9.1 Boundary walls, local boundary walls, global boundary walls.

Assume that an execution of Borcherds' method returned a set

$$\mathbb{D} = \{D_0, D_1, D_2, \ldots, D_r\}$$

of representatives of $\mathbf{H}$-congruence classes of chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$. We moreover assume that the conditions of corollary 31 hold, so that

$$\bigcup_{D \in \mathbb{D}} \mathbb{D} = D_0 \cup \cdots \cup D_r$$

is a fundamental domain of the action of $\mathrm{Aut}(X)$ onto $\mathrm{Nef}(X) \cap \mathcal{P}_S$. From now on, we will often refer to this fundamental domain as *the fundamental domain.* We now introduce the notions which will enable the reader to

- Produce graphical representations of the fundamental domain. These graphical representations are visually expressive and meaningful for cases where X has Picard number 3. Click here to view a few examples.

- Associate a cone to the fundamental domain and determine whether it is possible to compute its associated Hilbert basis.

To do so, it is important to characterize precisely the boundary of the fundamental domain.

Definition 32. A chamber $D \in \mathbb{D}$ is said to be *at the boundary* of the fundamental domain if there exists a chamber D' adjacent to D such that $D' \notin \mathbb{D}$.

Such a situation happens whenever there exists an element $m \in \Omega(D)$ such that the chamber adjacent to D along $(m)^\perp$ does not belong to $\mathbb{D}$. In this case, we say that $(m)^\perp$ is a boundary wall of the fundamental domain.

Definition 33. We say that a boundary wall $(m)^\perp$ is a *local boundary wall* if there exist chambers $D, D' \in \mathbb{D}$ adjacent to each other along $(m)^\perp$.

Definition 34. A boundary wall of the fundamental domain which is not a local boundary wall is called a *global boundary wall*

The facts exposed at the beginning of section 1.7 enable us to immediately deduce that (-2)-walls are by definition boundary walls since they form the boundary of

$$\mathrm{Nef}(X) \cap \mathcal{P}_S.$$

Recall that we denote by $\Omega^*(D)$ the set of non (-2)-walls of a $\mathcal{P}_S$-chamber D. Given a chamber $D \in \mathbb{D}$, we use the following procedure in order to identify boundary walls among the elements of $\Omega(D)$ and determine whether such walls are *local boundary walls* or a *global boundary walls*: Define initially empty sets

$$\mathcal{B}_{\text{dry}} = \{\,\} \qquad \text{and} \qquad \mathcal{L}_{\text{oc}} = \{\,\}.$$

Let $D \in \mathbb{D}$. For each $m \in \Omega^*(D)$, check whether the chamber D' adjacent to D along the wall $(m)^\perp$ belongs to $\mathbb{D}$ and proceed as follows:

- If $D' \in \mathbb{D}$, then $(m)^\perp$ is a boundary wall of the fundamental domain. In this case, we store the element m into $\mathcal{B}_{\text{dry}}$.

- If $D' \notin \mathbb{D}$ and the wall $(m)^\perp$ has already been identified has a boundary wall during the processing of another chamber of $\mathbb{D}$, i.e., $m \in \mathcal{B}_{\text{dry}}$, then $(m)^\perp$ is classified as a local boundary wall and stored into $\mathcal{L}_{\text{oc}}$.

Once all the chambers of $\mathbb{D}$ have thus been processed, then

- The set $\mathcal{B}_{\text{dry}}$ is the set of boundary walls.
- The set $\mathcal{L}_{\text{oc}}$ is the set of local boundary walls.
- The set
$$\mathcal{G}_{\text{lo}} = \mathcal{B}_{\text{dry}} \setminus \mathcal{L}_{\text{oc}}$$
is the set of global boundary walls.

Definition 35. The fundamental domain is said to be Hilbert Basis ready (*HB-ready*) whenever all its boundary walls are global boundary walls.

An *HB-ready* fundamental domain yields a convex polytope defined by the inequalities
$$\left\{x \in \mathbb{Q}^{\rho-1} \mid \text{for all } m \in \mathcal{B}_{\text{dry}},\ \langle x, m\rangle_{S^\vee} \geq 0\right\}$$
in $(\rho-1)$-dimensional space. SageMath features related to convex cones can be used to compute a Hilbert basis for the cone associated with this polytope. When the conditions of corollary 31 (that can be automatically checked by our implementation of Borcherds' method), our program **fundamentalizer** is capable of processing the data produced after an execution of Borcherds' method to carry out a study of the fundamental domain thus produced.

1.9.2 Graphical representation of the chamber structure of the fundamental domain.

In this section, we explain how to produce graphical representations in $(\rho-1)$-dimensional space of the fundamental domain of the action of $\mathrm{Aut}(X)$ onto

$\mathrm{Nef}(X) \cap \mathcal{P}_S$ which is produced by Borcherds' method when

$$\mathrm{Aut}_{\mathbf{H}}(D) = \{\mathrm{Id}\}$$

holds for all $D \in \mathbb{D}$. Let $D \in \mathbb{D}$ and $m \in \Omega(D)$. Assume that m is expressed in terms of its coordinates

$$m = [a_0, a_1, \ldots, a_{\rho-1}]_{S^\vee}$$

with respect to the basis of $S^\vee$. The principle enabling us to produce representations is straightforward: The wall

$$(m)^\perp = \{x \in S \otimes \mathbb{R} \mid \langle x, m\rangle_{S^\vee} = 0\} \cap \mathcal{P}_S$$

is associated with the hyperplane in $(\rho - 1)$-dimensional space defined as the solution set of the equation

$$a_0 + a_1x_1 + a_2x_2 + \ldots a_{\rho-2}x_{\rho-2} + a_{\rho-1}x_{\rho-1} = 0$$

This approach is particularly meaningful and visually unequivocal when the surface under study has Picard number 3: In this case, the wall associated with an element $[a_0, a_1, a_2]_{S^\vee} \in S^\vee$ is associated with the straight line defined by the equation $a_0 + a_1x + a_2y = 0$ in two-dimensional space. More precisely:

- If $a_2 \neq 0$ holds, then the wall defined by the orthogonal complement $[a_0, a_1, a_2] \in S^\vee$ is realized in to-dimensional space as an affine line with equation
$$y = -\frac{a_0}{a_2} - \frac{a_1}{a_2}x.$$
- If $a_2 = 0$ and $a_1 \neq 0$, then the wall is associated with the vertical line
$$x = -\frac{a_0}{a_1}.$$

1.10 Computing the (-2)-curves modulo Aut(X)

Assume that X is a $K3$ complex surface such that $\rho_X < 20$ and $-1 \notin \mathrm{Ker}(\eta_T)$. Theorem 22 then states that there is an isomorphism

$$\mathrm{Aut}(X) \simeq \mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S).$$

An execution of Borcherds' method thus provides a generating set for Aut(X) and outputs as complete set $\mathbb{D}$ of Aut(X)-congruence classes of chambers contained in Nef$(X) \cap \mathcal{P}_S$. If we assume moreover that

$$\mathrm{Aut}_{\mathbf{H}}(D) = \{\mathrm{Id}\}$$

holds for each $D \in \mathbb{D}$ then Corollary 31 then ensures that

$$\bigcup_{D \in \mathbb{D}} D$$

is a fundamental domain of the action of Aut(X) onto Nef$(X) \cap \mathcal{P}_S$. Borcherds' method also provides additional information regarding this fundamental domain: Recall that the set of walls $\Omega(D)$ of each chamber D explored by the method is processed by **RatDetect** to identify the (-2)-walls among it. Classes of smooth rational curves thus identified are then stored into the set $\mathcal{R}_{\mathrm{rat}}$ returned at the end of the execution of Borcherds' method. In this section, we establish two important facts:

- Each element of $\mathcal{R}_{\mathrm{rat}}$ represents an orbit of the set smooth rational curves on X under the action of Aut(X), and each such orbit possesses a representative in $\mathcal{R}_{\mathrm{rat}}$. The cardinality $\mathrm{Card}(\mathcal{R}_{\mathrm{rat}})$ of the set $\mathcal{R}_{\mathrm{rat}}$ therefore provides an upper bound on the number of orbits of smooth rational curves on X under the action of Aut(X).

Regarding the finiteness of the number of orbits of the set of smooth rational curves under the action of the automorphism group on $K3$ surfaces, we appeal to the following classical result due to Sterk [20]:

Theorem 36. *Let X be a $K3$ surface. The set of (-2)-curves up to automorphisms*

$$\left\{C \subset X \mid C \simeq \mathbb{P}^1\right\} \,/\, \mathit{Aut}(X)$$

is finite, i.e., there is a finite number of orbits of smooth rational curves.

This result is here stated in the form under which it can be found in Huybrechts' book [5], in which a proof is also provided. The second point addressed in this section consists in providing an operational template for an algorithmic method to refine the upper bound $\mathrm{Card}(\mathcal{R}_{\text{rat}})$.

- We will thus see that the upper bound $\mathrm{Card}(\mathcal{R}_{\text{rat}})$ on the number of orbits of smooth rational curves on X under the action of $\mathrm{Aut}(X)$ can be refined.

Indeed, the set $\mathcal{R}_{\text{rat}}$ can contain more than one representative for a given orbit. We thus provide an algorithmic solution to detect redundant representatives in $\mathcal{R}_{\text{rat}}$. A much more precise bound on the number of orbits will hence be obtained. Assume that

$$\mathcal{R}_{\text{rat}} = \{C_1, \ldots, C_s\}\,, \qquad \text{for some } s > 0$$

and let $C \in S$ be the class some smooth rational curve on X. We recall that have seen at the beginning of section 1.7 that no class of smooth rational curve is superfluous for cutting out $\mathrm{Nef}(X) \cap \mathcal{P}_S$. An immediate consequence of this fact is that there exists at least a $\mathcal{P}_S$-chamber

$$D \subset \mathrm{Nef}(X) \cap \mathcal{P}_S$$

having $(C)^\perp$ amongst its walls, i.e., such that $C \in \Omega(D)$. Two possibilities arise:

- If $D \in \mathbb{D}$, then C must be an element of $\mathcal{R}_{\text{rat}}$, i.e., C must have been detected by Borcherds' method during its execution.

- If $D \notin \mathbb{D}$, then the fact that $\mathbb{D}$ is a complete set of representatives of $\mathrm{Aut}(X)$-congruence classes of chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$ enables us to

assert that there exists a transformation

$$g \in \mathrm{Aut}(X)$$

such that

$$D^g = D' \in \mathbb{D}.$$

In this case, the transformation g thus sends D onto a chamber $D' \in \mathbb{D}$. Recall that the class $C \in S$ is assumed to be the class of a smooth rational curve on X. Since an automorphism sends the class of a smooth rational curve onto the class of a smooth rational curve, the image of the class $C \in S$, by the transformation g must be sent to an element of $\mathcal{R}_{\text{rat}}$. We therefore have

$$C^g \in \mathcal{R}_{\text{rat}}.$$

We thus established the following proposition:

Proposition 37. *Assume that X satisfies the conditions of Theorem 22 and that $Aut_{\mathbf{H}}(D) = \{Id\}$ holds for every $D \in \mathbb{D}$. Each orbit of the set of smooth rational curves on X possesses at least one representative contained in the set $\mathcal{R}_{rat}$.*

The set $\mathcal{R}_{\text{rat}}$ may, however, contain more than one representative of orbits. Denote by $\mathcal{S}_{\text{orb}}$ the set of orbits of smooth rational curves on X under the action of $\mathrm{Aut}(X)$. Proposition 37 then implies that

$$\mathrm{Card}\left(\mathcal{S}_{\text{orb}}\right) \leq \mathrm{Card}(\mathcal{R}_{\text{rat}}) \tag{1.26}$$

so that $\mathrm{Card}(\mathcal{R}_{\text{rat}})$ is an upper bound on the number of orbits of smooth rational curves. We now explain how this upper bound can be refined. We start by reviewing the means which could be used as leverage to do so. Denote by

$$\mathcal{O}(C) = \{C^g \mid g \in \mathrm{Aut}(X)\}$$

the orbit of a class $C \in S$ of a curve on X under the action of $\mathrm{Aut}(X)$. For any

two distinct elements $C_i, C_j \in \mathcal{R}_{\text{rat}}$ such that

$$\mathcal{O}(C_i) \neq \mathcal{O}(C_j),$$

we have

$$\mathcal{O}(C_i) \cap \mathcal{O}(C_j) = \emptyset.$$

since the very definition of an orbit which implies that any two distinct orbit must have an empty intersection. Before proceeding further, let us discuss practical considerations: Note that $K3$ surfaces with finite automorphism groups have already been studied in detail, a wealth of information on these surfaces can be found in Roulleau's atlas of $K3$ surfaces with finite automorphism group [16]. We, therefore, focus on $K3$ surfaces with infinite automorphism group. For such surfaces, orbits of elements of $S = \mathrm{NS}(X)$ under the action of $\mathrm{Aut}(X)$ are by definition infinite sets. It is thus impossible to explicitly compute the orbit $O(C)$ of any element $C \in S$. Our computer-based algorithmic approach is indeed bound by the fact that we must confine ourselves to dealing with **finite** objects. Taking this fact into account, we now reformulate what we just discussed in terms of finite sets: Assume given distinct elements $C_i, C_j \in S$. Then it is clear that the assumption

$$\mathcal{O}(C_i) \neq \mathcal{O}(C_j)$$

implies that for all subsets

$$A \subseteq \mathcal{O}(C_i) \qquad \text{and} \qquad B \subseteq \mathcal{O}(C_j)$$

we have

$$A \cap B = \emptyset.$$

Taking the contrapositive of this implication leads us to the fact that finding finite subsets

$$A \subseteq \mathcal{O}(C_i) \qquad \text{and} \qquad B \subseteq \mathcal{O}(C_j)$$

satisfying

$$A \cap B \neq \emptyset$$

is enough in order to establish the equality

$$\mathcal{O}(C_i) = \mathcal{O}(C_j).$$

We thus introduce the notion of *partial orbit*: Given a finite subset

$$\mathrm{Aut}_{\mathrm{par}}(X) \subset \mathrm{Aut}(X),$$

the partial orbit of an element $C \in S$ is the finite subset of $\mathcal{O}(C)$ defined as

$$\mathcal{O}_{\mathrm{par}}(C) = \left\{ C^g \mid g \in \mathrm{Aut}_{\mathrm{par}}(X) \right\}.$$

We then recall that Borcherds' method returns a generating set Γ of $\mathrm{Aut}(X)$ and define

$$\Gamma^* = \Gamma \cup \left\{ g^{-1} \mid g \in \Gamma \right\} \cup \{\mathrm{Id}\},$$

to be the extended generating set obtained by adding inverses and the identity to Γ. In order to refine the upper bound $\mathrm{Card}(\mathcal{R}_{\mathrm{rat}})$ on the number of orbits of smooth rational curves under the action of $\mathrm{Aut}(X)$,we proceed by enforcing the following procedure :

Upper bound refinement procedure:

- We compute a finite subset $\mathrm{Aut}_{\mathrm{par}}(X)$ of $\mathrm{Aut}(X)$.
- We then use this subset to compute the partial orbit

$$\mathcal{O}_{\mathrm{par}}(C) = \left\{ C^g \mid g \in \mathrm{Aut}_{\mathrm{par}}(X) \right\}$$

  for each $C \in \mathcal{R}_{\mathrm{rat}}$ and form the set of partial orbits of elements of $\mathcal{R}_{\mathrm{rat}}$.

For each $C \in \mathcal{R}_{\mathrm{rat}}$ we then proceed as follows: For each $C' \in \mathcal{R}_{\mathrm{rat}} \setminus \{C\}$:

- If $\mathcal{O}_{\text{par}}(C) \cap \mathcal{O}_{\text{par}}(C') \neq \emptyset$ then clearly $\mathcal{O}(C) = \mathcal{O}(C')$ as discussed earlier in this section. Either one of C or C' is then removed from $\mathcal{R}_{\text{rat}}$ so that the upper bound is decreased by 1.

- If $\mathcal{O}_{\text{par}}(C) \cap \mathcal{O}_{\text{par}}(C') = \emptyset$ then for each element $g \in \Gamma^* \setminus \{\text{Id}\}$ we compute the sets

  $$\mathcal{O}_{\text{par}}(C)^g = \left\{xg \mid x \in \mathcal{O}_{\text{par}}(C)\right\} \text{ and } \mathcal{O}_{\text{par}}(C')^g = \left\{xh \mid x \in \mathcal{O}_{\text{par}}(C')\right\}$$

  and determine whether there exist elements $g, h \in \Gamma^* \setminus \{\text{Id}\}$ such that

  $$\mathcal{O}_{\text{par}}(C)^g \cap \mathcal{O}_{\text{par}}(C')^h \neq \emptyset$$

  When this is the case, then $\mathcal{O}(C) = \mathcal{O}(C')$. As before, the upper bound is then decreased by 1 and either one of C, C' is removed from the set $\mathcal{R}_{\text{rat}}$.

Assume that this procedure has been executed and that $\mathcal{R}_{\text{rat}}$ has thus been updated. That is, we assume that the procedure returned an updated set $\mathcal{R}'_{\text{rat}} \subseteq \mathcal{R}_{\text{rat}}$. Then for any two distinct elements $C, C' \in \mathcal{R}'_{\text{rat}}$ and any two distinct elements $g, h \in \Gamma^* \setminus \{\text{Id}\}$, the equality $\mathcal{O}_{\text{par}}(C)^g \cap \mathcal{O}_{\text{par}}(C')^h = \emptyset$ holds. We can thus assert the non-existence of elements of $\text{Aut}(X)$ acting as a non-trivial permution on the set of partial orbits of the elements of $\mathcal{R}'_{\text{rat}}$. We then take $\text{Card}(\mathcal{R}'_{\text{rat}})$ as our refined upper bound on the number of orbits of smooth rational curves on X under the action of $\text{Aut}(X)$ and consider than no further refinement can be easily obtained from the data of $\text{Aut}_{\text{par}}(X)$. In case we desire to refine the upper bound further, we need to compute a more extensive set of elements of $\text{Aut}(X)$ and apply the above procedure again. We now explain how we proceed in order to compute finite subsets $\text{Aut}_{\text{par}}(X) \subset \text{Aut}(X)$. Our procedure to do so is motivated by the fact that, as far as we know, there is currently no computer-based solution in public access that takes as input a set of generators of an infinite group G, an integer $p > 0$, and outputs a set of elements of this group having cardinality equal to p. Fix a strictly positive integer p. We

show how to explicitly compute a subset $\mathrm{Aut}_{\mathrm{par}}(X) \subseteq \mathrm{Aut}(X)$ such that

$$\mathrm{Card}(\mathrm{Aut}_{\mathrm{par}}(X)) \geq p,$$

that is, a subset having cardinality at least equal to p.

Procedure AutParGen: Denote by

$$\mathcal{W}(\Gamma^*, \beta) = \{a_1 \dots a_\beta \mid a_i \in \Gamma^* \qquad 1 \leq i \leq \beta\}$$

the set of words of length less than or equal to β in the free group over the set Γ^*. Obiously, we have $\mathcal{W}(\Gamma^*, \beta) \subset \mathrm{Aut}(X)$ no matter the value of β. Recall that Γ^* has been as defined as $\Gamma \cup \Gamma^{-1}$ minus the identity. Thus, there are strict inclusions $\mathcal{W}(\Gamma^*, \gamma) \subset \mathcal{W}(\Gamma^*, \beta)$ whenever $\alpha < \beta$, with α and β positive integers. Also, note that $\mathrm{Card}(\mathcal{W}(\Gamma^*, \beta)) \leq \mathrm{Card}(\Gamma^*)^\beta$ holds. Before proceeding further, we recall that the floor and ceiling functions are both functions taking real values as input and returning integers defined as

$$\mathrm{floor} : x \in \mathbb{R} \longmapsto \max\{m \in \mathbb{Z} \mid m \leq x\}$$

and

$$\mathrm{ceiling} : x \in \mathbb{R} \longmapsto \min\{m \in \mathbb{Z} \mid x \leq m\}.$$

Denote by β_0 be the **greatest** integer N such that $\mathrm{Card}(\Gamma^*)^N \leq p$, that is:

$$\beta_0 = \mathrm{floor}(\frac{\log(p)}{\log(\mathrm{Card}\,(\Gamma^*))})$$

We then compute $\mathcal{W}(\Gamma^*, \beta_0)$, which , as indicated above, satisfies

$$\mathrm{Card}(\mathcal{W}(\Gamma_{\mathrm{ext}}, \beta_0)) \leq p.$$

In order to reach our goal of obtaining a finite subset of $\mathrm{Aut}(X)$ having cardi-

nality superior or equal to p, we still have to compute at least

$$p - \mathrm{Card}(\mathcal{W}(\Gamma^*, \beta))$$

additional elements of $\mathrm{Aut}(X)$. To do so, we compute a sequence of sets $\mathcal{W}_j$:

- Define $\mathcal{W}_0$ as a copy of $\mathcal{W}(\Gamma_0, \beta^*)$.

Assume that $\mathcal{W}_j$ has been computed and proceed as follows:

- If $\mathrm{Card}(\mathcal{W}_j) \geq p$ holds, then the goal of obtaining a subset of $\mathrm{Aut}(X)$ of cardinality at least equal to p has been achieved, the procedure stops.
- Otherwise, we compute $\mathcal{W}_{j+1}$. We start by defining

$$\delta_{j+1} = \mathrm{ceiling}(\frac{p}{\mathrm{Card}(\mathcal{W}_j)}).$$

  There are two possibilities:

  - If $\delta_{j+1} < \mathrm{Card}(\Gamma^*)$, pick a subset $\mathcal{S}_{j+1} \subset \Gamma^* \setminus \{\mathrm{Id}\}$ such that

$$\mathrm{Card}(\mathcal{S}_{j+1}) = \delta_{j+1}.$$

  - Otherwise, define $\mathcal{S}_{j+1}$ as a copy of Γ^*.

  We then compute the set

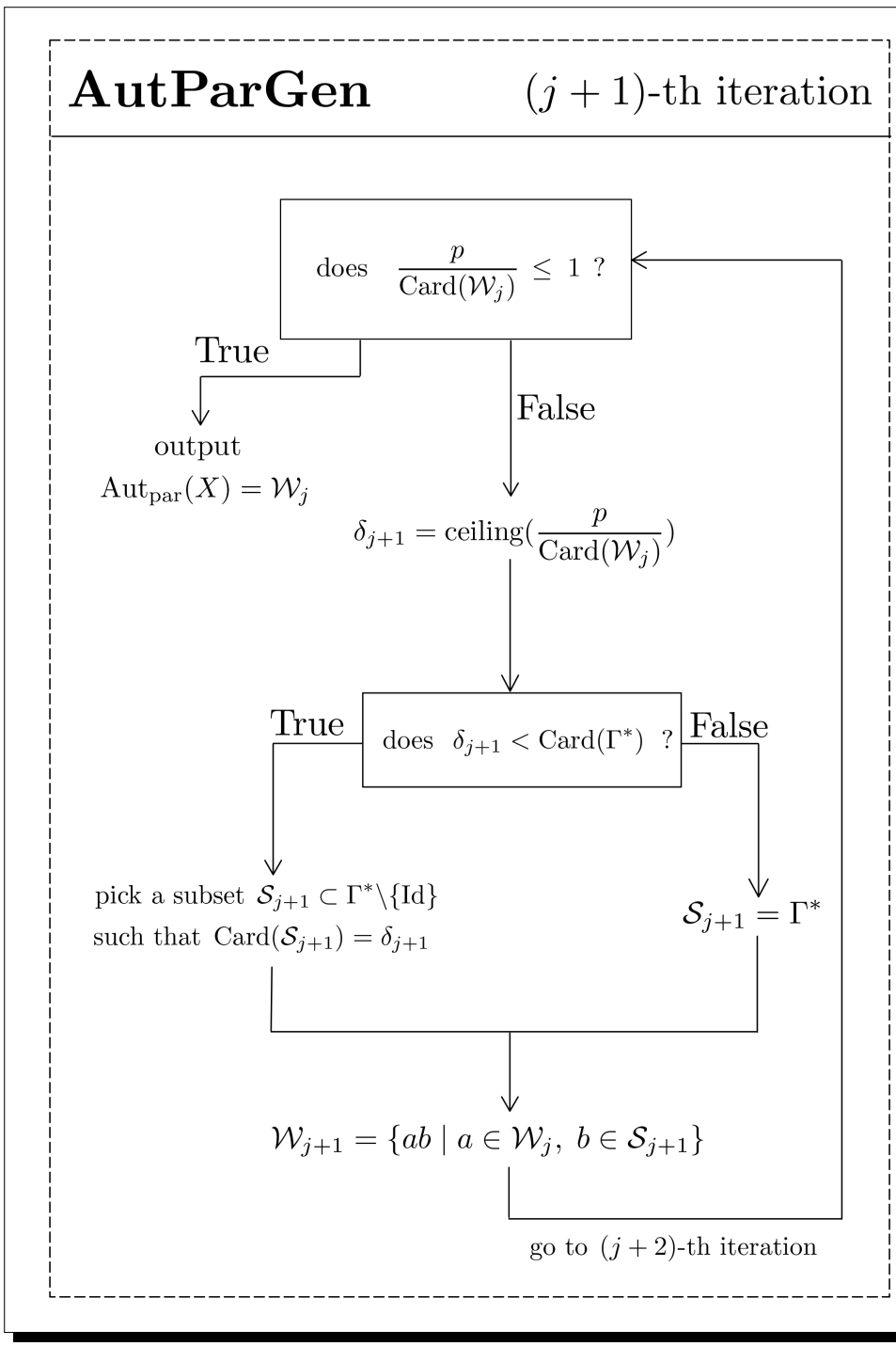
$$\mathcal{W}_{j+1} = \{ab \mid a \in \mathcal{W}_j,\ b \in \mathcal{S}_{j+1}\}$$

  and go back at the beginning of this procedure with $\mathcal{W}_{j+1}$ as input data.

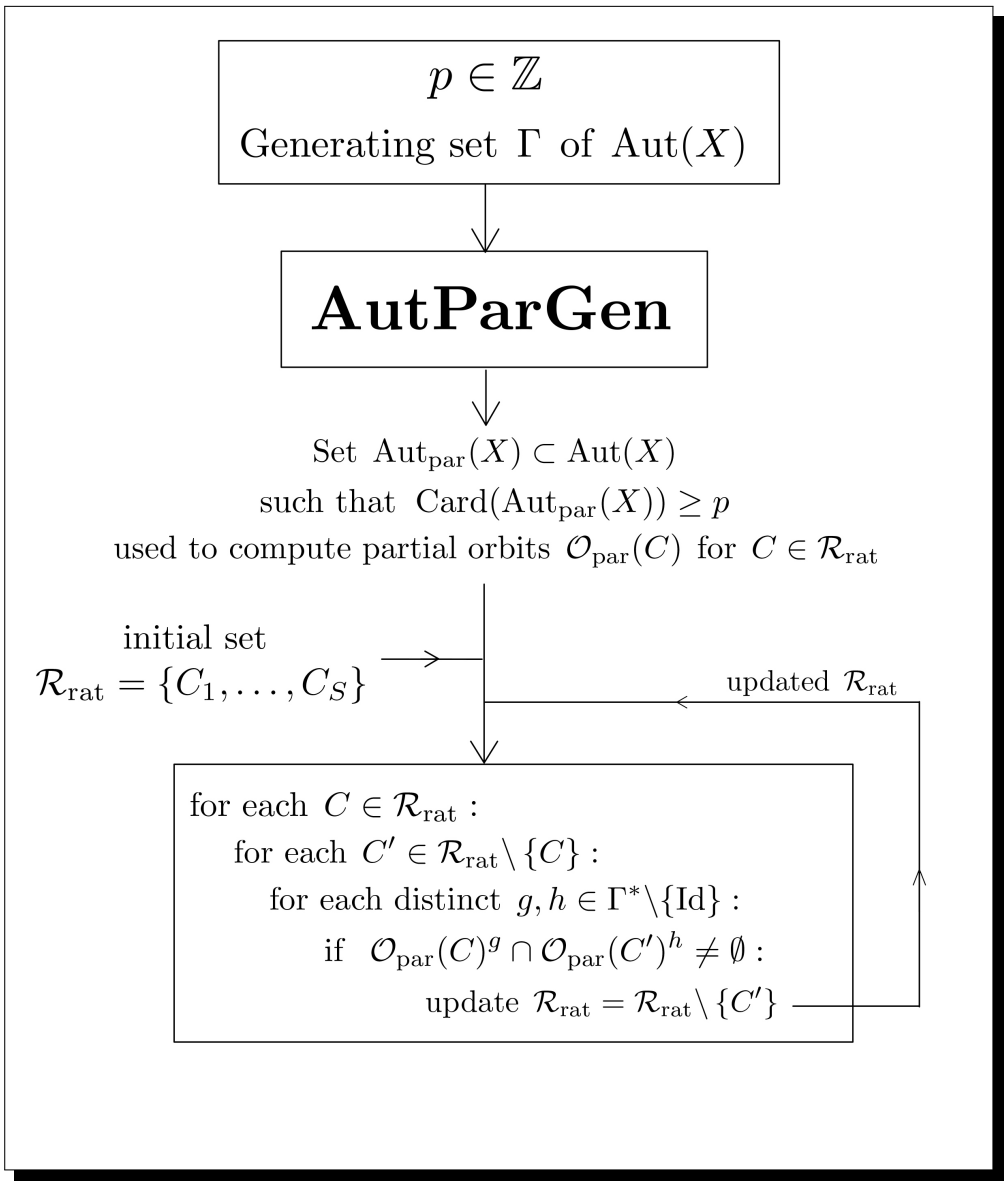
After a few iterations, a set $\mathcal{W}_m$ satisfying $\mathrm{Card}(\mathcal{W}_m) \geq p$ will be obtained for some integer m. We set

$$\mathrm{Aut}_{\mathrm{par}}(X) = \mathcal{W}_m.$$

The structure of the procedure **AutParGen** can be summarized as follows :


AutParGen
$(j+1)$-th iteration
does $\frac{p}{\mathrm{Card}(\mathcal{W}_j)} \leq 1$?
True
False
output
$\mathrm{Aut}_{\mathrm{par}}(X) = \mathcal{W}_j$
$\delta_{j+1} = \mathrm{ceiling}(\frac{p}{\mathrm{Card}(\mathcal{W}_j)})$
True
does $\delta_{j+1} < \mathrm{Card}(\Gamma^*)$?
False
pick a subset $\mathcal{S}_{j+1} \subset \Gamma^* \backslash \{\mathrm{Id}\}$
such that $\mathrm{Card}(\mathcal{S}_{j+1}) = \delta_{j+1}$
$\mathcal{S}_{j+1} = \Gamma^*$
$\mathcal{W}_{j+1} = \{ab \mid a \in \mathcal{W}_j,\ b \in \mathcal{S}_{j+1}\}$
go to $(j+2)$-th iteration

The whole upper bound refinement procedure introduced in this section can be schematized as follows:



1.11 Toward a parallelized Borcherds' method

In order to compute a generating set of $\mathrm{Aut}(X)$, Borcherds' method enforces means which are **brute force** flavored, by design. As discussed in section 1.7.4 and even mentioned by Shimada himself in his article [19], the mechanics on which relies the congruence testing procedure **CongChecker** testify to this fact. Using **brute force**, however, has a price in terms of resources and computation times. Due to the large amount of data that has to be processed depending on the $K3$ under study, executing Borcherds' method may require time. For instance, when computing a generating set of $\mathrm{Aut}(X_t)$ for a $K3$ surface X_t with Picard number 5, i.e., having a Néron-Severi group with Gram matrix

$$\begin{bmatrix} 2t & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & -2 \end{bmatrix}$$

with respect to a fixed basis, we observed that whenever $t \geq 5$ Borcherds' method has to deal with more than 80.000 representatives of congruence classes of chambers during the final stages of its execution. Since each newly explored chamber has to be tested for congruency against each such representative, the method has to perform tens of thousands of congruence tests for each newly discovered chamber. When $t \geq 7$, the number of representatives is way over 100.000. Our idea to deal with this issue is based on common sense principles: We modernized the method in such a way that procedures such as congruence testing can be deployed in parallel over various worker processes. Let us use an example to illustrate this idea : Assume that Borcherds' method is exploring a chamber D and that this chamber has to be tested for congruency against the elements of the set

$$\mathcal{S} = \{D_0, D_1, \ldots, D_{79999}\}$$

That is, Borcherds' method has to apply **CongChecker** 80.000 times: First, D has to be tested against D_0, then against D_1, then against D_2,..., and finally tested for congruency against D_{79999}. The congruence testing part of the classical Borcherds' method, as described by Shimada in his article [19], almost a decade ago, was intended to be implemented over a single **for** loop, i.e.,

for each chamber

$$D_k \in \{D_0, D_1, \ldots, D_{79999}\}$$

run **CongChecker**(D, D_k).

We cannot abide by such an old-fashioned approach in 2022. Common sense dictates that instead of performing 80.000 congruence tests in series, this workload should be split over, say, 16 processes P_j, $1 \leq j \leq 16$, in parallel, where each process is expected to perform $1/16$th of the overall workload, that is, 5000 congruence tests. Formally, we take the set $\mathcal{S}$ and split it into 16 subsets $\mathcal{S}_j$ for $j \in \{1, 2, \ldots, 16\}$ of cardinality 5000. We here assume that a machine having a CPU with at least 16 logical cores is available, thus enabling the OS scheduler to dispatch each of these 16 worker processes over a dedicated core for parallel execution, making the best possible use of the CPU resources available. In fact, things are quite simple : Enforcing this approach amounts to running 16 **for** loop in parallel : Each loop iterates 5000 times, instead of a single **for** loop iterating 80000 times.

(Process P_j)
for each chamber

$$D' \in \mathcal{S}_j,$$

run **CongChecker**(D, D').

One remark : In an online section, we explain that Shimada's approach to congruence testing can be massively modernized. These enhancements, combined with parallel deployment of congruence testing, enabled us to obtain astonishing results for cases involving a hefty number of representatives of cong. classes.

The Python **multiprocessing** package includes a major asset which suits perfectly our needs: The **Pool** object. As indicated in the Python official documentation, the **Pool** object offers a convenient means of parallelizing the execution of a function across multiple input values, distributing the input data across processes (data parallelism). We thus made use of the **Pool** object to parallelize many procedures within our implementation of Borcherds' method, thus achieving massive performance improvements compared to our early implementations of Borcherds' method produced by following to the letter the guidelines from Shimada's that can be found in [19]. We expand on this matter in the section 1.11 of this thesis. Since we always try to make the most out of the hardware at our disposal, we have to mention that our first attempt to speed up our implementation of Borcherds' method consisted in using GPU computing to perform the matrix multiplications which occur during an execution of Borcherds' method. However, the small size (at most 26×26) of the matrices involved in Borcherds' method does not allow GPU computing to express all its power. Our experiment initially consisted in managing to be able to perform CUDA operations in Sage's Python environment, with an old RX580. Guidelines on how to reproduce this experiment are provided on the website K3surfaces.com. We ended by setting the GPU approach aside and focused on parallelism involving CPU computing for the remainder of our study. In this section, we proceed as follows:

- In section 1.11.1, we start by introducing the basic principles behind process-based parallelism. We then provide a quick overview of the internal procedures of Borcherds' method under the viewpoint of parallel deployment, focusing on those suitable for doing so. We then introduce the structure of a modernized version of Borcherds' method : The *Poolized Borcherds' method*, which arises due to the enforcement of process-based parallelism with the **Pool** object from **multiprocessing**.
- In section 1.11.2, we explain how we applied parallelism at the scale of Borcherds' method itself, thus opening new concrete perspectives.

1.11.1 The Poolized Borcherds' method

Assume that a massive pile of sand has to be cleared from the entrance of a town building. A city worker arrives in front of the disaster and can either clear the pile on its own or bring reinforcements by calling his colleagues and mobilizing a team of municipal workers. In practice, a team of workers should be much more efficient than a single individual in clearing a huge pile of sand. Working efficiently as a team, however, requires coordination. To this end, tasks must be precisely defined and distributed evenly across the team of workers, which are assumed to be endowed with equal abilities. Assume that the pile of sand is the finite set

$$\mathcal{S} = \{q_1, q_2, \ldots, q_M\},$$

with M integer. Assume that clearing the pile of sand consists in applying a procedure f to the elements of this set. Mobilizing only one worker to clear the pile of sand amounts to performing

$$f(q_1),\ f(q_2), \ldots,\ f(q_M)$$

in series. One worker, alone and on his own, digs out the sand, shovel by shovel. Depending on the size of the pile, the clearing process may take a lot of time. An analogous situation on a computer would be the execution of a single process, i.e., of a sequence of tasks, performed one by one, in series, sequentially, on a single core at any given time. On the other hand, mobilizing a team of workers and evenly splitting the workload between all the team members will reduce the amount of time required to clear the pile of sand. On a computer, the sequence of tasks to be accomplished would then be distributed over more than one process, running in parallel and making the best possible use of available resources.

Note that clearing a pile of sand with a team made of various workers can be done without defining any particular order. Distributing the work between available workers is enough: **Order does not matter**. Tasks for which order is irrelevant should thus be considered first when enforcing parallelism.

Regarding Borcherds' method, what are the tasks for which the order does not count? Taking a look at the figure depicting the algorithmic structure of the method, an obvious candidate stands out: **Congruence Testing**, that is, the procedure **CongChecker**. Indeed, when a newly explored chamber D has to be tested for congruency against each of the representatives of congruences classes of chambers previously discovered. The order in which the tests are performed does not matter. Similarly, when **RatDetect** is applied to the set of walls of a chamber, wall by wall, order does not count. Another example is the computation of the Weyl vectors of chambers adjacent to a given chamber along its non-(-2)-walls (w.r.t. anti-backtracking) : The order to which **WeylAdj** is applied to these walls is irrelevant. We thus have already identified at least three internal procedures of Borcherds' method which are obvious and suitable candidates for deployment in parallel over various worker processes. Congruence testing, due to its computationally intensive brute-force nature, is the one for which we have the most to gain by enforcing process-based parallelism, e.g., by deploying **CongChecker** in parallel over various worker processes. Assume that N worker processes

$$P_1, \ldots, P_N$$

can be mobilized in parallel, each and that some procedure f has to be applied to each element of a finite set $\mathcal{S}$. We assume that the overall result of the whole operation is not impacted by the order to which the elements of $\mathcal{S}$ are processed. How should we proceed to distribute the workload of having to process the elements of $\mathcal{S}$ to take advantage of our N worker processes?

- If $\mathrm{Card}(\mathcal{S}) < N$, we call for a number $\mathrm{Card}(\mathcal{S})$ of workers and assign one element of $\mathcal{S}$ to each of them. The other $N - \mathrm{Card}(\mathcal{S})$ workers idle.
- If $\mathrm{Card}(\mathcal{S}) \geq N$ then we split $\mathcal{S}$ into N subsets $\mathcal{S}_1, \ldots, \mathcal{S}_N$ and assign each of them to a worker $\mathcal{S}_j \longmapsto P_j$ which applies the procedure f to each element of $\mathcal{S}_j$.

This is as simple as it looks.

In order to determine the cardinality $\mathrm{Card}(\mathcal{S}_j)$ of each of the N subsets $\mathcal{S}_j \subset \mathcal{S}$ in such a way that

$$\sum_{j=1}^{N} \mathrm{Card}(\mathcal{S}_j) = \mathrm{Card}(\mathcal{S}),$$

we can proceed as follows. We start by performing the euclidean division of $\mathrm{Card}(\mathcal{S})$ by N. There exist integers q and r with $r < N$ such that

$$\mathrm{Card}(\mathcal{S}) = Nq + r.$$

Moreover, the assumption $r < N$ enables us to find an integer $\delta_{N,r}$ such that

$$N = r + \delta_{N,r}.$$

We can thus express $\mathrm{Card}(\mathcal{S})$ as:

$$\begin{aligned} \mathrm{Card}(\mathcal{S}) &= (r + \delta_{N,r})q + r \\ &= (q+1)r + q\delta_{N,r}. \end{aligned}$$

A natural way of partitioning $\mathcal{S}$ into $N = r + \delta_{N,r}$ subsets thus appears.

The set $\mathcal{S}$ is split into:

- r subsets each containing $q + 1$ elements.
- $\delta_{N,r}$ subsets each containing q elements.

Example. Assume that Borcherds' method is at the beginning of its 53th iteration during the computation of a generating set of $\mathrm{Aut}(X_5)$ where X_5 is the $K3$ with Picard number five in the case where $t = 5$, with Gram matrix presented at the beginning of section 1.11. The first newly discovered chamber D has to be tested for $\mathbf{H}$-congruency against 80.218 representatives of congruence classes. Thus, we have $\mathrm{Card}(\mathcal{S}) = 80128$. Assume that we have a CPU with at least 16 logical cores, so that we can execute 16 worker processes in parallel without hassle. We set $N = 16$. Proceeding as above, we have

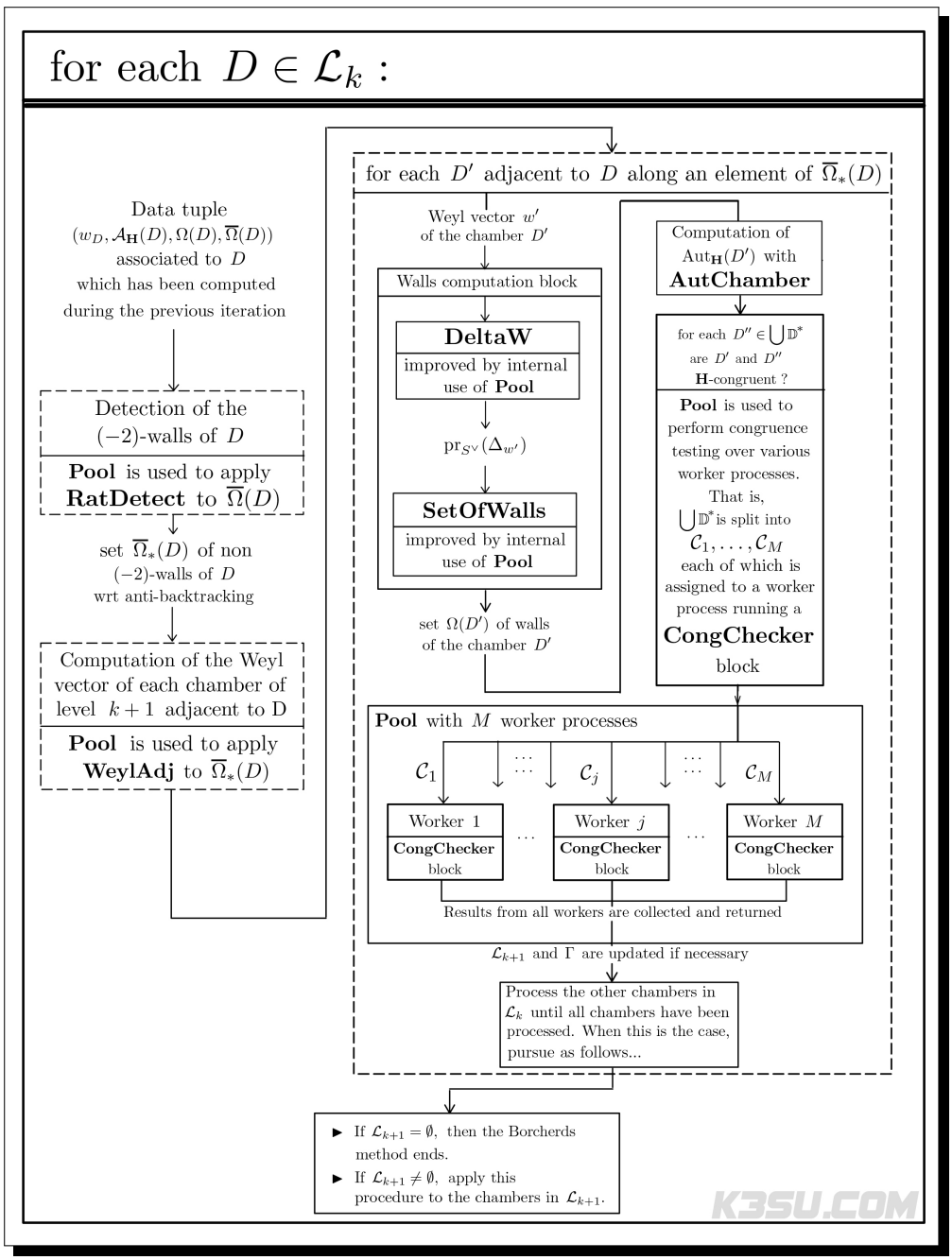
$$\begin{aligned} 80218 &= 5013 \cdot 16 + 10 \\ &= 5013 \cdot (6 + 10) + 10 \\ &= 5013 \cdot 6 + 5014 \cdot 10 \end{aligned}$$

We thus split the workload into 6 subsets having each cardinality 5013, and 10 other subsets each having cardinality 5014.

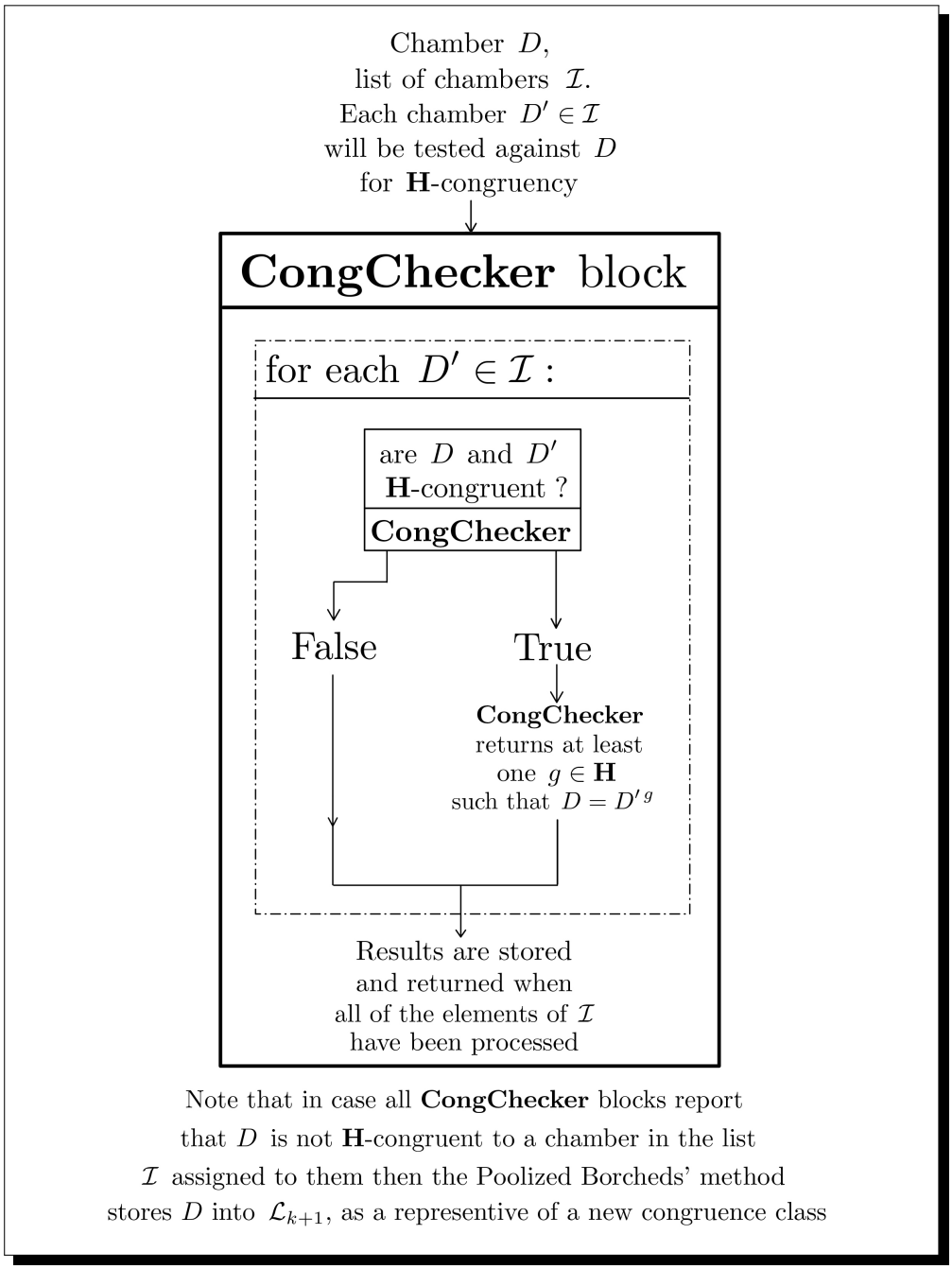
We introduced the main ideas and basic concepts behind process-based parallelism. We now introduce the tools that enabled us to enforce these concepts with a computer-based approach. The **Pool** object from the Python **multiprocessing** library is an efficient, flexible and reliable tool that can be used to make a Python program benefit from the use of process-based parallelism, thus enabling us to take full advantage of the multi-core architecture of a CPU. Adapting our code in order to take advantage of the **Pool** object was worth the effort. The key do so consists of having a clear and global view of the algorithmic structure under study. Critical points that can benefit from deployment of procedures in parallel can then be readily identified. Going into precise details on how we made use of **Pool** and produced the Poolized Borcherds' method is a battle that cannot be fought in this PDF, but can be fought online. Nevertheless, we will provide a few bits of advice and explain where we used **Pool** within Borcherds' method. First, we had to identify the tasks and procedures which can lead to noticeable performance gains when deployed in parallel.

For instance, the **for** loops in which time and resource expensive functions are repeatedly executed are usually critical spots on which performance gains can be obtained by enforcing process based-parallelism. Within Borcherds' method, we used process-based parallelism at the level of the procedures mentioned below. Note that doing so led us to the Poolized Borcherds method.

- **Detection of (-2)-walls:** In order to identify the (-2)-walls among the elements of the set of walls of a chamber, the procedure **RatDetect** described in section 1.7.1 can be deployed in parallel.
- **Computation of Weyl vectors:** The procedure **WeylAdj** can be deployed in parallel to compute the Weyl vector of each chamber adjacent to a chamber along its non (-2)-walls.
- **Computation of the set of walls:** The procedure **DeltaW** repeatedly calls the Shimada's custom ShortVectors algorithm **ShiVectors** (described in section 1.4. These calls can be distributed over various worker processes. Within **SetOfWalls**, the chunk of code involving **linprog** from **scipy.optimize**, used to deal with Shimada's LP problem, is deployed in parallel over various worker processes, through **Pool**.
- **Congruence tests:** As discussed previously, Borcherds' method may have to perform congruence tests by repeatedly applying the procedure **CongChecker** described in section 1.7.4. Tens of thousands of congruence tests may have to be performed for each newly discovered chamber. It, therefore, makes perfect sense to distribute the resulting workload over multiple worker processes, each running a **CongChecker** block and processing a chunk of the total workload. We took great care in optimizing and enhancing for execution in series Shimada's 2013 congruence testing procedure before proceeding to parallel deployment. **Parallel deployment should not be done blindly.** We recommend at least trying to get the most out of the procedures, deployed in series, and then starting thinking parallel. Doing so enabled us to obtain substantial gains.


for each $D \in \mathcal{L}_k$:
Data tuple $(w_D, \mathcal{A}_{\mathbf{H}}(D), \Omega(D), \overline{\Omega}(D))$ associated to D which has been computed during the previous iteration
Detection of the (-2)-walls of D
Pool is used to apply RatDetect to $\overline{\Omega}(D)$
set $\overline{\Omega}_*(D)$ of non (-2)-walls of D wrt anti-backtracking
Computation of the Weyl vector of each chamber of level $k+1$ adjacent to D
Pool is used to apply WeylAdj to $\overline{\Omega}_*(D)$
for each D' adjacent to D along an element of $\overline{\Omega}_*(D)$
Weyl vector w' of the chamber D'
Walls computation block
DeltaW
improved by internal use of Pool
$\mathrm{pr}_{S^\vee}(\Delta_{w'})$
SetOfWalls
improved by internal use of Pool
set $\Omega(D')$ of walls of the chamber D'
Computation of $\mathrm{Aut}_{\mathbf{H}}(D')$ with AutChamber
for each $D'' \in \bigcup \mathbb{D}^*$ are D' and D'' H-congruent ?
Pool is used to perform congruence testing over various worker processes. That is, $\bigcup \mathbb{D}^*$ is split into $\mathcal{C}_1, \ldots, \mathcal{C}_M$ each of which is assigned to a worker process running a CongChecker block
Pool with M worker processes
$\mathcal{C}_1$
$\mathcal{C}_j$
$\mathcal{C}_M$
Worker 1 CongChecker block
Worker j CongChecker block
Worker M CongChecker block
Results from all workers are collected and returned
$\mathcal{L}_{k+1}$ and Γ are updated if necessary
Process the other chambers in $\mathcal{L}_k$ until all chambers have been processed. When this is the case, pursue as follows...
If $\mathcal{L}_{k+1} = \emptyset$, then the Borcherds method ends.
If $\mathcal{L}_{k+1} \neq \emptyset$, apply this procedure to the chambers in $\mathcal{L}_{k+1}$.
K3SU.COM

The following figure illustrates the structure of a **CongChecker** block:



Note that in case all **CongChecker** blocks report that D is not **H**-congruent to a chamber in the list $\mathcal{I}$ assigned to them then the Poolized Borcheds' method stores D into $\mathcal{L}_{k+1}$, as a representive of a new congruence class

1.11.2 Enforcing parallelism at the scale of Borcherds' method

We introduced the basic principles of process-based parallelism. These principles enabled us to implement Borcherds' method while taking advantage of modern hardware. We now explain the most straightforward way to make use of parallelism on a broader scale. Instead of using process-based parallelism inside of the method, that is, at the level of its internal procedures, we will use it at the level Borcherds' method itself. We first have to remember that the method relies on two core components to fulfill its ultimate purpose:

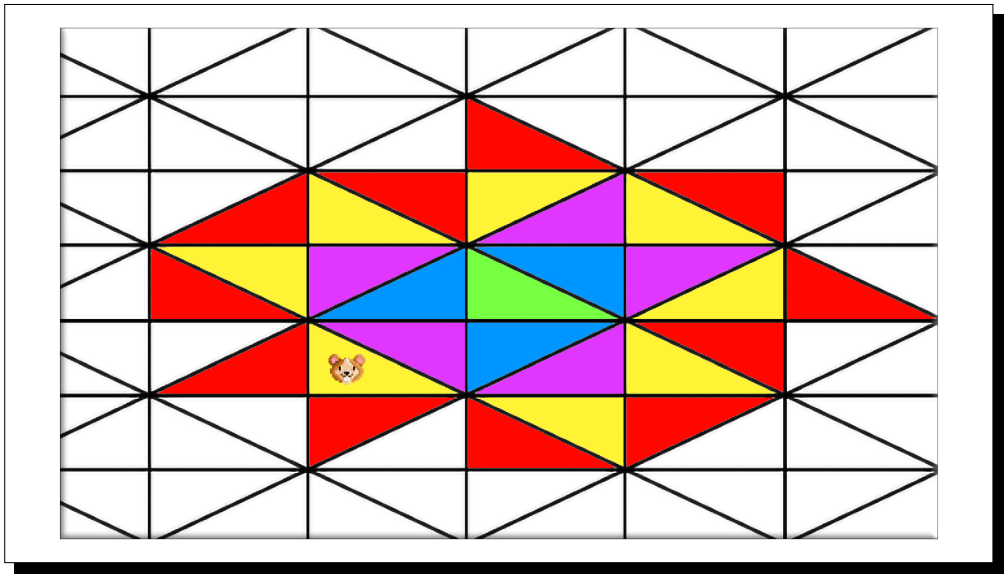
- One component enables the method to **explore** the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$.
- The other enables the method to **process** this chamber structure in order to obtain generators of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$.

Using process-based parallelism within the method enables us to obtain massive improvements on the **processing** component of Borcherds' method. Indeed, we have seen in the previous section that a solution such as **Pool** can be used to deploy in parallel or enhance procedures such as **CongChecker**, **DeltaW** or **RatDetect**. We now focus on using parallelism on the exploration component of Borcherds' method. Going back to the end of section 1.7 and taking a look at the figure depicting the structure of the classical Borcherds' method, it is clear that the backbone of the algorithmic structure of Borcherds' method is a top-level **for** loop: For each chamber $D \in \mathcal{L}_k$, the set of level-k representatives of congruence classes of chambers, Borcherds' method discovers chambers of level $k+1$ by exploring the surroundings of D along its non (-2)-walls. On paper, the most straightforward course of action consists in distributing the workload represented by $\mathcal{L}_k$ over various worker processes. Let's visualize this idea by using a picture. We keep things simple : During its execution, Borcherds' method can be viewed as a hamster exploring a chamber structure. The following figure is based on a genuine representation of a chunk of the chamber structure over $\mathrm{Nef}(X) \cap \mathcal{P}_S$ when X is the $K3$ surface with Néron-Séveri $S = \mathrm{NS}(X)$ having

Gram matrix equal to

$$\begin{bmatrix} 84 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

with respect to a fixed basis.



Assume that :

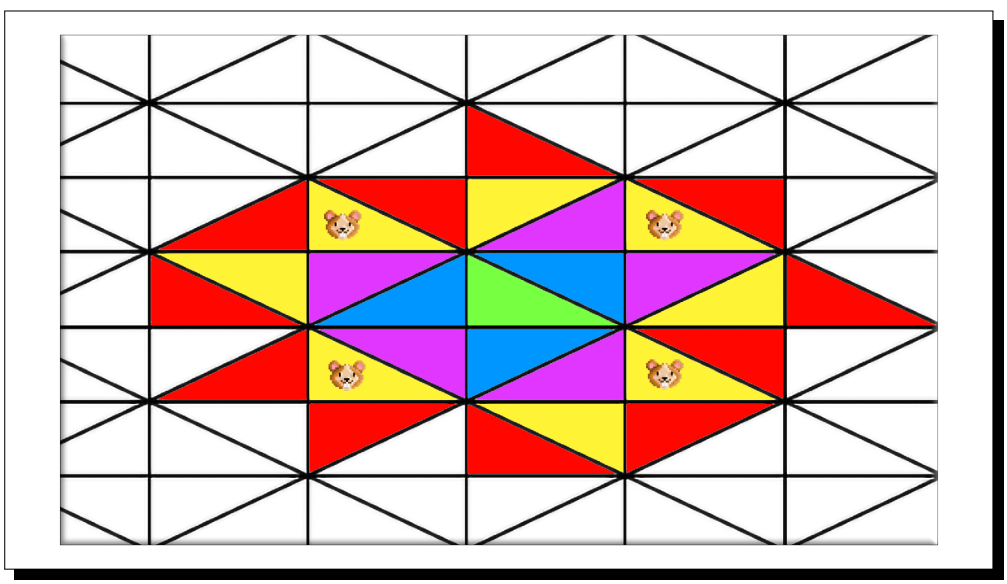
- The green-colored chamber is the initial chamber.
- Chambers in blue are chamber of level 1.
- Chambers in purple are chambers of level 2.
- Chambers in yellow are chambers of level 3.
- Chambers in red are chambers of level 5.

Assume that Borcherds' method is represented by the hamster emoji, as pictured above. Furthermore, we assume that the method starts exploring and processing chambers of level 4, colored in red, by adjacency to chambers of level 3, in yel-

low. Please note that we assume that each chamber in yellow represents its own congruence class of chambers, i.e., we assume that $\mathcal{L}_3$ contains all the chambers in yellow. Our idea merely consists in distributing over multiple processes the workload represented by exploring and computing the walls of chambers adjacent to chambers in $\mathcal{L}_3$. For example, we can split $\mathcal{L}_3$, which contains the 8 yellow chambers, into four subsets

$$\mathcal{L}_3^{(1)}, \mathcal{L}_3^{(2)}, \mathcal{L}_3^{(3)}, \mathcal{L}_3^{(4)}.$$

each containing 2 chambers. We then assign each of these subsets $\mathcal{L}_k^{(j)}$ to a process, each represented by a hamster Emoji, in charge of exploring and processing red chambers adjacent to the chambers in $\mathcal{L}_k^{(j)}$. We illustrate the situation by updating our previous figure :



Each one of the four hamsters pictured above would thus receive an assignment of two yellow chambers, and would have to explore and process their adjacencies. In practice, many issues arise when such a straightforward parallelized approach of Borcherds' method is implemented. If we assume that each hamster is capable of fully enforcing the features of Borcherds' method, we have to

avoid and mitigate the consequences of the two following issues :

Issue n°1 - The pitfall of unrestricted nested parallelism: Assume that we launch in parallel N instances of a program capable of enforcing all the internal procedures of Borcherds' method. Assume that each such instance can deploy these features with a process-based parallelism solution such as the **Pool** object. That is, we assume each instance can mobilize its own dedicated team of workers, e.g., M dedicated worker processes, that can be mobilized to deploy procedures such as **RatDetect**, **WeylAdj** or **CongChecker** with process-based parallelism. We then have to keep a firm eye on resources. The question is then : Is our machine powerful enough to handle a total number of $M \times N$ resource-hungry processes running in parallel? Taking modest values such as $N = M = 4$ already yields a total of 16 processes, each potentially mobilizing the full power of a logical core. We could be facing a CPU bottleneck situation. Due to the state of technology when this thesis was produced, such a situation would then have been a severe issue for most consumer-grade machines. We have to carefully pick the values of M and N to efficiently allocate the available resources and thus obtain the best performance ratio.

Issue n°2 - Communication is necessary to work efficiently as a team: Assume that the burden of exploring and processing chambers of level $k+2$ by adjacency to chambers of level $k+1$ has been distributed over various processes. We thus assume that $\mathcal{L}_{k+1}$ has been split into N subsets $\mathcal{L}_{k+1}^{(j)}$ each assigned to a worker process P_j which will explore and process the chambers of level $k+1$ adjacent to chambers in $\mathcal{L}_{k+1}^{(j)}$ along their non (-2)-walls for $1 \leq j \leq N$. Assume that

- A chamber $D_1 \in \mathcal{L}_{k+1}^{(1)}$ is discovered by process P_1 at time t_1.
- A chamber $D_2 \in \mathcal{L}_{k+1}^{(2)}$ is discovered by process P_2 at time $t_2 > t_1$.
- D_1 and D_2 are distinct and congruent.

Process P_1 computes the set of walls of D_1 and test it for congruence against all the representatives of congruence classes of chambers which are already known. Assume that D_1 represents a new congruence class. We have to make sure that processes share information and communicate through a shared database. Indeed, Process P_2 needs to be informed of the existence of D_1 so that the chamber D_2 it discovers can be tested against D_1 for congruency. The chamber D_2 will otherwise be classified as representing a new congruence class. A dramatic consequence of this situation is that a new generator of $\mathrm{Aut}_{\mathbf{H}}(\mathrm{Nef}(X) \cap \mathcal{P}_S)$ which could have been obtained by testing D_2 against D_1 for congruency with **CongChecker** could here remain undiscovered forever, thus skewing the purpose, intent, and results of the execution of Borcherds' method. We thus see that enforcing parallelism at the scale of Borcherds' method cannot be done while ignoring the issue of communication between processes. Indeed, some tasks imperatively require communication between processes, as we just discussed in the case of congruence testing. We did not have enough time to produce an implementation of Borcherds' method involving various processes capable of performing their own congruence tests while being synchronized and communicating through a common database. We, however, urge people to go in this direction in the future. Enabling processes to conduct their own congruence tests while sharing data in real-time is undoubtedly one of the significant challenges regarding the future of Borcherds' method. We must however concede that in order to deal with most surfaces with small Picard number, using parallelism at the internal scale of a single instance of Borcherds' method, e.g., enforcing congruence testing over a pool of 16 or 20 worker processes is more than enough to complete an execution of the method in a reasonable amount of time. Indeed, the cardinality of the complete set of representatives of congruence classes of chambers of $\mathrm{Nef}(X) \cap \mathcal{P}_S$ obtained at the end of an execution of Borcherds' method on such surfaces does not usually exceed a few thousand chambers, at most. As indicated on K3surfaces.com, generating sets of the respective automorphism groups of various famous surfaces with Picard number 3 or 4 can be obtained in a matter of seconds, minutes at worst, and yield a com-

plete set of representatives of small cardinality. Regarding the less straightforward cases, our extensive use of process-based parallelism at the internal level of the method, combined with a substantial preliminary effort to optimize the procedures themselves nevertheless enabled us to obtain significant improvements. For instance, during an application of the method on a $K3$ of Picard number five, which involved more than tens of thousands of representatives of congruence classes, we observed that testing a given chamber for congruency against 80231 other chambers was 1000 times faster with our modernized approach than when we used our programs implemented by following to the letter Shimada's guidelines from his 2013 article [19]. Despite these improvements, we observed that a severe limiting factor in terms of computation times still had to be considered and put under control: The computation time of the sets of walls of a chamber. Our idea to deal with this issue consisted in enforcing parallelism at the level of Borcherds' method itself, as discussed earlier, but this time with an approach focused on the parallel deployment of various processes to perform the exploration of the chamber structure and of the computation of the sets of walls of chambers. To this end, we adopted a strategy based on the use of a **primary** process P_0 and of **auxiliary** processes $P_1, \ldots, P_N$, as follows:

- The primary process P_0 is a full instance of Borcherds' method.
- Worker processes $P_1, \ldots, P_N$ are endowed with Borcherds' method features **RatDetect** and **WeylAdj** to navigate within the chamber structure and of **DeltaW** and **SetOfWalls** to compute the respective sets of walls of chambers.
- All processes are synchronized by level and communicate through a common shared database.
- All processes are allowed to deploy their internal procedures using process-based parallelism to accomplish their duties.

The mechanics behind this approach can be described as follows: At the beginning of each iteration, say the $(k+2)$-th iteration, the set $\mathcal{L}_{k+1}$ containing

the chambers whose adjacencies are to be explored and processed is split into N subsets $\mathcal{L}_{k+1}^{(j)}$ for $j \in \{1, \ldots, N\}$. We set $\mathcal{L}_{k+1}^{(0)} = \mathcal{L}_{k+1}$ and assign to each process P_j :

- A subset $\mathcal{L}_{k+1}^{(j)} \subseteq \mathcal{L}_{k+1}$
- A set $\mathcal{R}_{k+1}^{(j)}$ containing the sets of walls which will be computed by P_j.
- A set $\mathcal{E}_{k+1}^{(j)}$ containing the data of the Weyl vector of chambers whose set of walls have been computed by P_j.

As soon as the process P_j has computed the Weyl vector w_D of a chamber D of level $k+2$ adjacent to a chamber in $\mathcal{L}_{k+1}^{(j)}$ along a non (-2)-wall , P_j checks whether the condition

$$w_D \in \bigcup_{i=0}^{N} \mathcal{E}_{k+1}^{(i)} \tag{1.27}$$

holds. That is, the auxiliary process P_j checks whether D has already been explored and processed earlier by another process. Two possibilities then arise:

- If the boolean value associated with the expression (1.27) is **true**, then the process P_j either proceeds to its next task in line or idles until the next iteration if P_j has already completed the exploration of the adjacencies of all chambers in its assigned share of the workload $\mathcal{L}_{k+1}^{(j)}$.
- If the boolean value associated with the expression (1.27) is **false**, then P_j knows that D has never been explored before, and thus takes care of the computation of the set of walls of D. As soon as P_j completes the computation of $\Omega(D)$, it stores a copy of this set into $\mathcal{R}_{k+1}^{(j)}$, and stores a copy the Weyl vector w_D of D into the set $\mathcal{E}_{k+1}^{(j)}$ so that other processes can know that $\Omega(D)$ has indeed been already been computed by P_j during the $(k+2)$-th iteration, and should not be computed again.

We hence see that the two main purposes of the auxiliary processes P_j consist in *(a)* exploring the chambers adjacent to chambers in their respective assigned share $\mathcal{L}_{k+1}^{(j)}$ of the entire workload $\mathcal{L}_{k+1}$, and *(b)* computing the respective sets of walls of these chambers if these sets have not been computed earlier.

Auxiliary processes $P_j \neq P_0$ thus work to the benefit of the primary process P_0. The latter is a full instance of the **Poolized** Borcherds' method. Whenever the mechanics of Borcherds' method would require the primary process P_0 to compute the set of walls $\Omega(D)$ of a $\mathcal{P}_S$-chamber D with Weyl vector w_D, the impact of our new approach lies in the fact that P_0 now checks whether

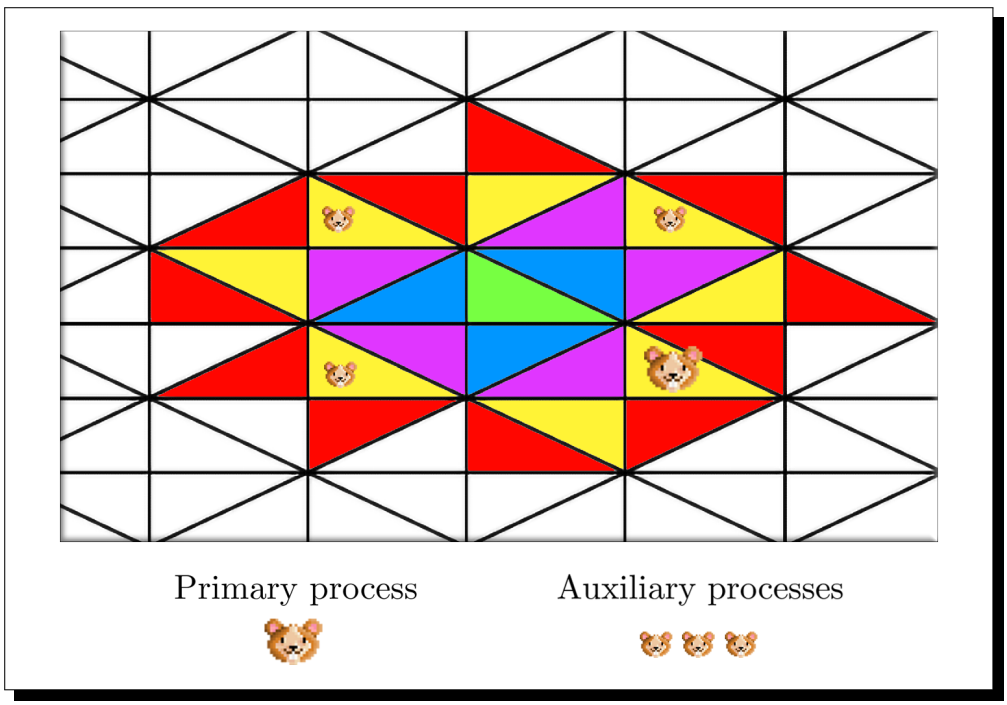
$$w_D \in \bigcup_{i=0}^{N} \mathcal{E}_{k+1}^{(i)}$$

holds. When this is the case, the primary process P_0 retrieves $\Omega(D)$ from

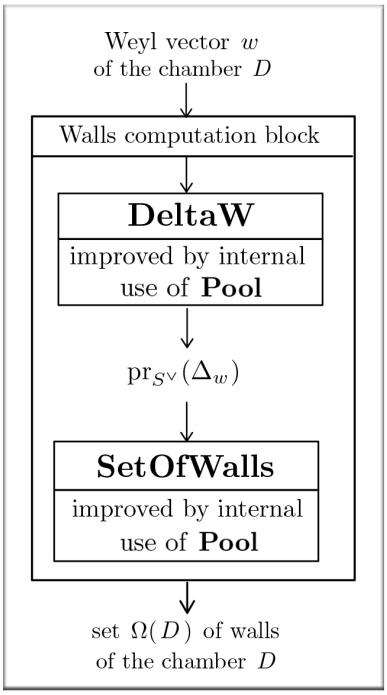
$$\bigcup_{i=0}^{N} \mathcal{R}_{k+1}^{(i)}$$

and thus does not have to spend time and resources on the computation of this set of walls. Otherwise, the primary process P_0 computes the set of walls $\Omega(D)$ of D, stores a copy of $\Omega(D)$ into $\mathcal{R}_{k+1}^{(0)}$, and a copy of w_D into $\mathcal{E}_{k+1}^{(0)}$.

We devised this strategy in such a way that Borcherds' method can be fully executed by the primary process P_0 no matter what auxiliary processes produce. Even if the execution of all the auxiliary processes $P_j \neq P_0$ is interrupted, the primary process P_0 can thus continue running Borcherds' method all by itself. The situation is illustrated in the following figure.



We thus represent the primary process as a giant hamster. The path of this giant hamster inside the chamber structure does not depend on the behavior of the tiny hamsters. During the iteration, the giant hamster explores each of the red chambers adjacent to the yellow chambers. However, the tiny hamsters, which represent auxiliary processes, work in sync to the benefit of the primary process. These smaller hamsters compute the sets of walls of red chambers, which have been assigned specifically to each of them at the beginning of the iteration by the giant hamster, and thus enable the latter to have direct access to the data of these sets of walls when needed, thus minimizing the workload over the giant hamster's shoulders in terms of the computation of sets of walls. We now formally explain how we enforced this approach. In order to enable the primary process P_0 to communicate with the auxiliary processes through a shared database, we swap the **Walls computation block**, from the Poolized Borcherds' method, also displayed on the following page,



for a new functional block, called the **P**oolized **F**unctional **B**lock. A **PFB** block takes a Weyl vector w_D of a chamber D as input, determines whether the set of walls of D has already been computed by a worker process, and computes its set of walls whenever this is not the case. Hence, a **PFB** block can test whether the condition

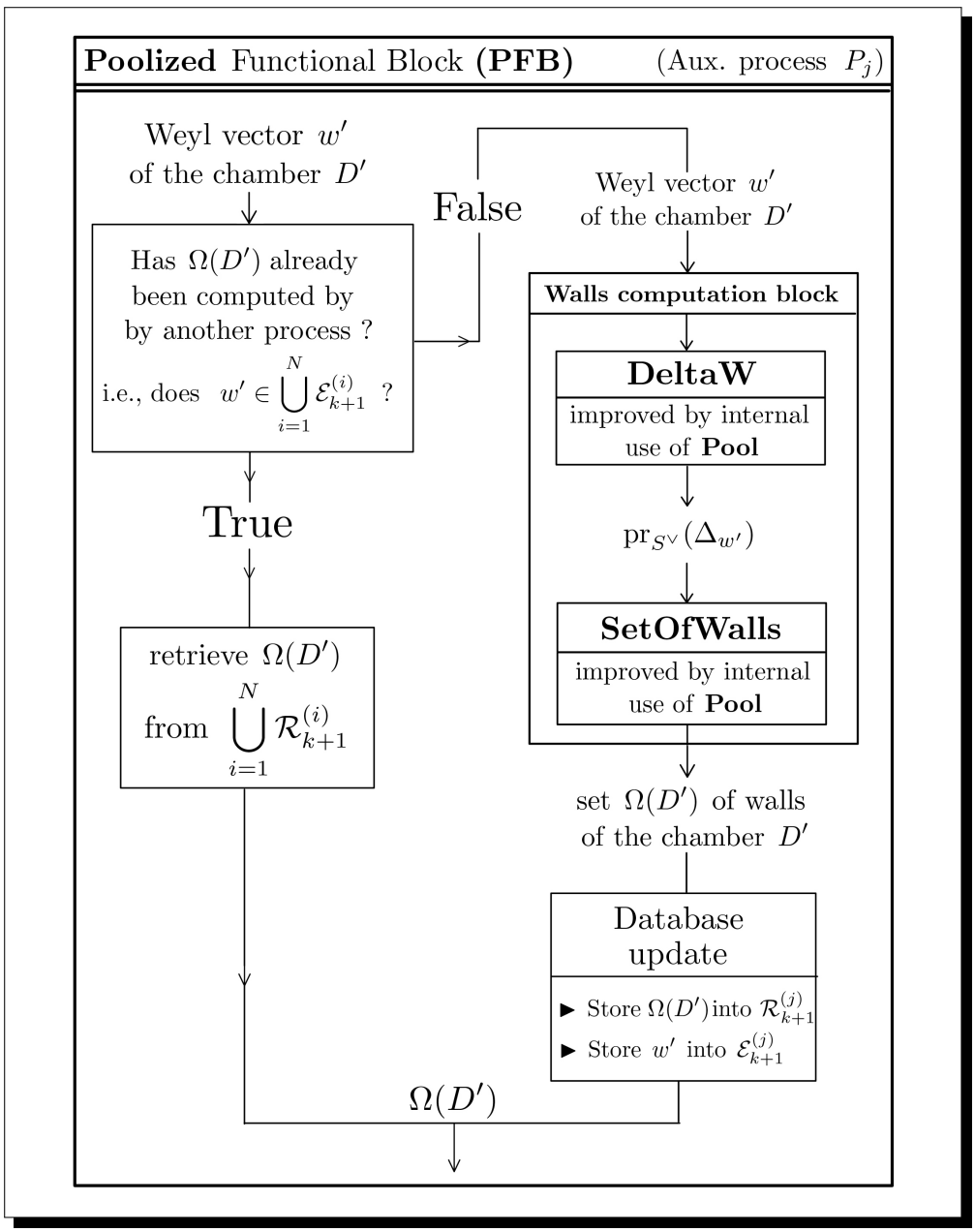
$$w_D \in \bigcup_{i=0}^{N} \mathcal{E}_{k+1}^{(i)}$$

holds. As we already discussed, two possibilities then arise:

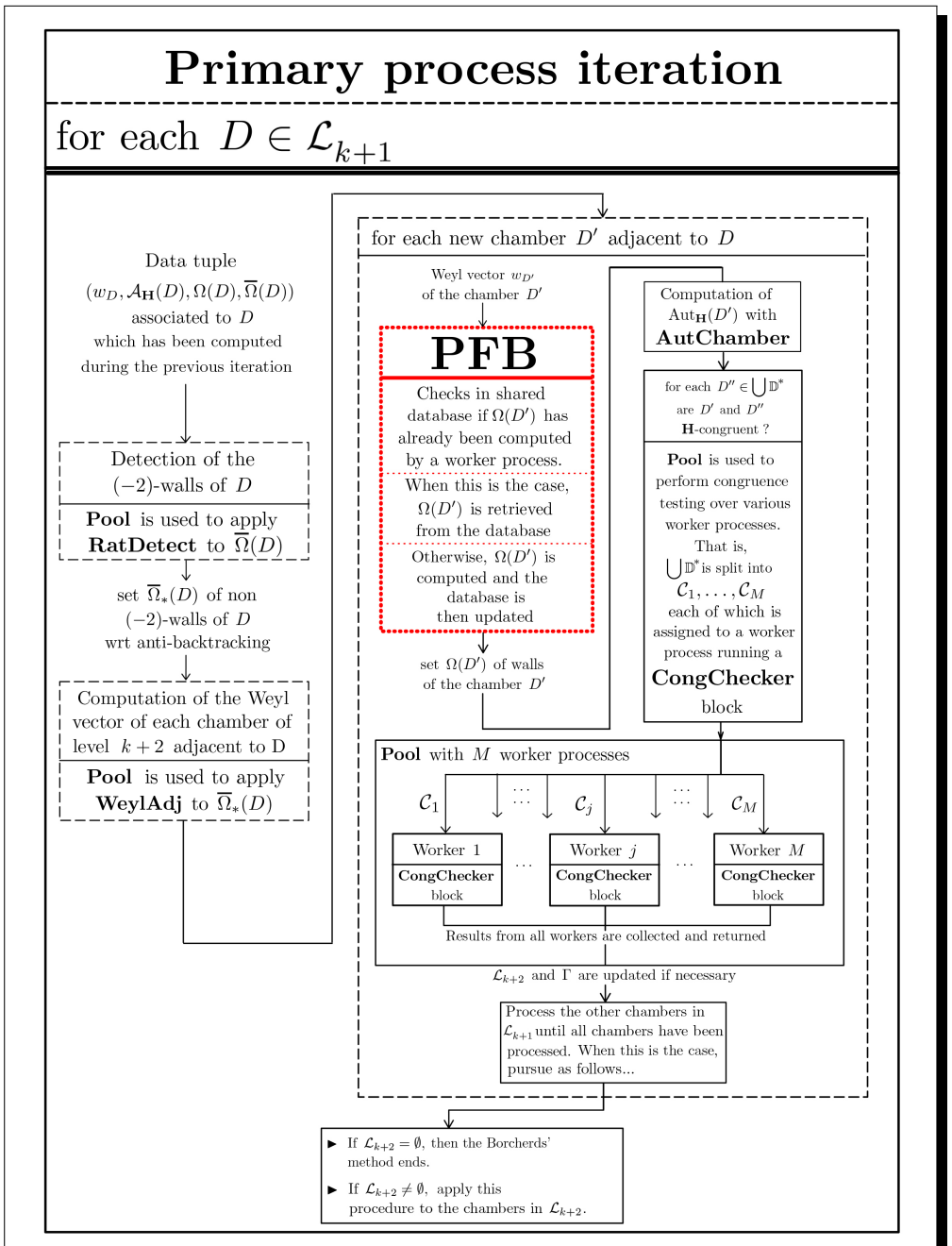
- Whenever this condition holds, **PFB** retrieves the data of $\Omega(D)$ from

$$\bigcup_{i=0}^{N} \mathcal{R}_{k+1}^{(i)}.$$

- The **PFB** block otherwise computes $\Omega(D)$ with **DeltaW** and **SetOfWalls**.



The inner workings of a **PFB** block are depicted in the figure above, while the updated algorithmic structure of the **Poolized** Borcherds' method augmented with a **PFB** block is illustrated in the following figure.


Primary process iteration
for each $D \in \mathcal{L}_{k+1}$
Data tuple
$(w_D, \mathcal{A}_{\mathbf{H}}(D), \Omega(D), \overline{\Omega}(D))$
associated to D
which has been computed
during the previous iteration
Detection of the
(-2)-walls of D
Pool is used to apply
RatDetect to $\overline{\Omega}(D)$
set $\overline{\Omega}_*(D)$ of non
(-2)-walls of D
wrt anti-backtracking
Computation of the Weyl
vector of each chamber of
level $k+2$ adjacent to D
Pool is used to apply
WeylAdj to $\overline{\Omega}_*(D)$
for each new chamber D' adjacent to D
Weyl vector $w_{D'}$
of the chamber D'
PFB
Checks in shared
database if $\Omega(D')$ has
already been computed
by a worker process.
When this is the case,
$\Omega(D')$ is retrieved
from the database
Otherwise, $\Omega(D')$ is
computed and the
database is
then updated
set $\Omega(D')$ of walls
of the chamber D'
Computation of
$\mathrm{Aut}_{\mathbf{H}}(D')$ with
AutChamber
for each $D'' \in \bigcup \mathbb{D}^*$
are D' and D''
H-congruent ?
Pool is used to
perform congruence
testing over various
worker processes.
That is,
$\bigcup \mathbb{D}^*$ is split into
$\mathcal{C}_1, \ldots, \mathcal{C}_M$
each of which is
assigned to a worker
process running a
CongChecker
block
Pool with M worker processes
$\mathcal{C}_1$
$\mathcal{C}_j$
$\mathcal{C}_M$
Worker 1
CongChecker
block
Worker j
CongChecker
block
Worker M
CongChecker
block
Results from all workers are collected and returned
$\mathcal{L}_{k+2}$ and Γ are updated if necessary
Process the other chambers in
$\mathcal{L}_{k+1}$ until all chambers have been
processed. When this is the case,
pursue as follows...
If $\mathcal{L}_{k+2} = \emptyset$, then the Borcherds'
method ends.
If $\mathcal{L}_{k+2} \neq \emptyset$, apply this
procedure to the chambers in $\mathcal{L}_{k+2}$.

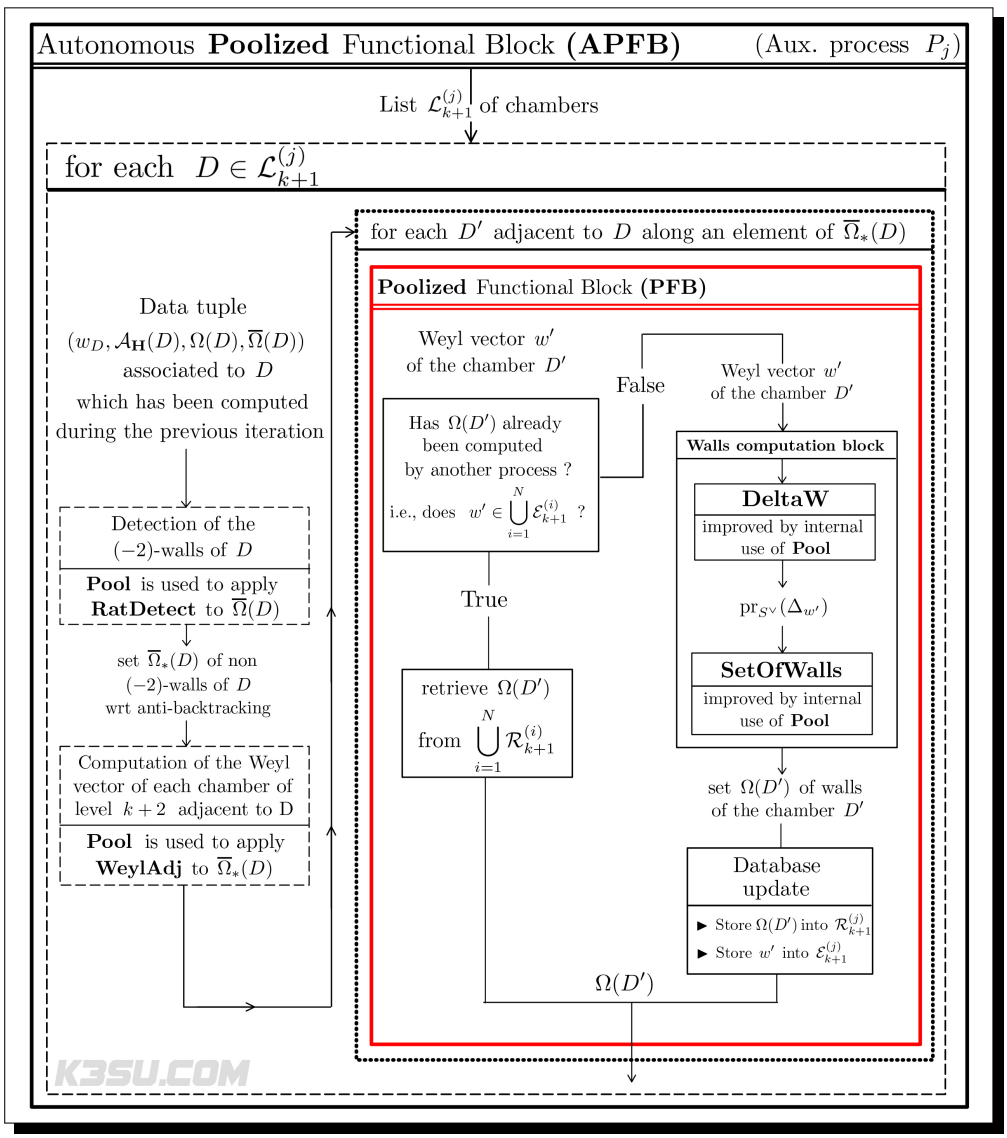
We still have to precisely formalize the mechanics behind the auxiliary processes $P_j \neq P_0$. Before proceeding further, recall that we assume that Borcherds' method is starting its $(k+2)$-th iteration and that the primary process P_0 splits $\mathcal{L}_{k+1}$ into subsets

$$\mathcal{L}_{k+1}^{(1)}, \ldots, \mathcal{L}_{k+1}^{(N)}$$

each assigned to an auxiliary process P_j. Such a auxiliary process P_j must be able to:

- Identify (-2)-walls among the walls of chambers in $\mathcal{L}_{k+1}^{(j)}$ so that chambers of level $k+1$ adjacent along such walls will not be visited. Thus, auxiliary processes must be able to execute the procedure **RatDetect**.
- Compute the Weyl vector of the chambers of level $k+1$ adjacent to chambers in $\mathcal{L}_{k+1}^{(j)}$ along their non (-2)-walls. Auxiliary processes, therefore, need to include the procedure **WeylAdj** among their features.
- Consult the shared database to determine whether the set of walls of a given chamber has already been computed.
- Compute the set of walls of a chamber so that the procedures **DeltaW** and **SetOfWalls** have to be among the procedures that can be executed by auxiliary processes.

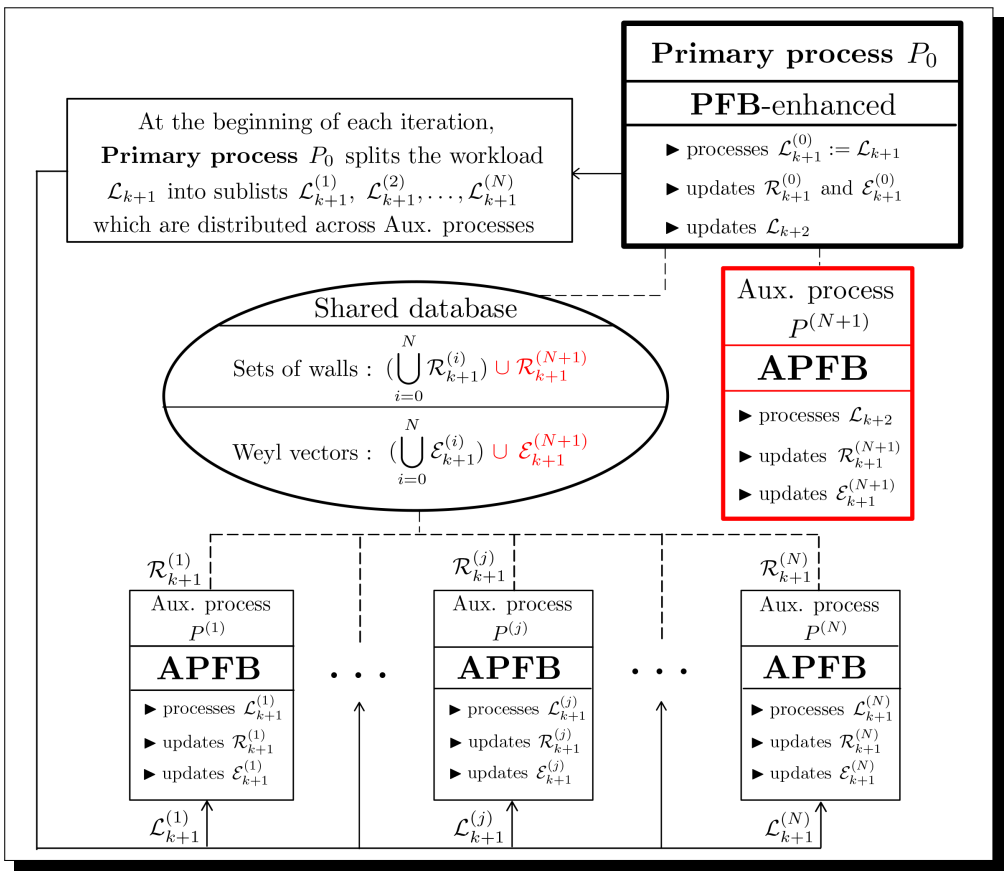
We formalize these requirements by introducing an enhanced version of the **PFB** block, called the **A**utonomous **P**oolized **F**unctional **B**lock, or **APFB**. This block is obtained by combining **RatDetect** and **WeylAdj** to a **PFB** block, thus making the latter autonomous by enabling it to safely navigate within the portion of the chamber structure assigned to the auxiliary process over which it is executed. An important thing to remember is that both **PFB** and **APFB** can deploy their respective internal procedures using process-based parallelism with **Pool**, hence the **P** in their respective abbreviated names, for **P**oolized.


Autonomous **Poolized** Functional Block **(APFB)**
(Aux. process P_j)
List $\mathcal{L}_{k+1}^{(j)}$ of chambers
for each $D \in \mathcal{L}_{k+1}^{(j)}$
for each D' adjacent to D along an element of $\overline{\Omega}_*(D)$
Poolized Functional Block **(PFB)**
Data tuple
$(w_D, \mathcal{A}_{\mathbf{H}}(D), \Omega(D), \overline{\Omega}(D))$
associated to D
which has been computed
during the previous iteration
Detection of the
(-2)-walls of D
Pool is used to apply
RatDetect to $\overline{\Omega}(D)$
set $\overline{\Omega}_*(D)$ of non
(-2)-walls of D
wrt anti-backtracking
Computation of the Weyl
vector of each chamber of
level $k+2$ adjacent to D
Pool is used to apply
WeylAdj to $\overline{\Omega}_*(D)$
Weyl vector w'
of the chamber D'
Has $\Omega(D')$ already
been computed
by another process ?
i.e., does $w' \in \bigcup_{i=1}^{N} \mathcal{E}_{k+1}^{(i)}$?
False
True
Weyl vector w'
of the chamber D'
Walls computation block
DeltaW
improved by internal
use of **Pool**
$\mathrm{pr}_{S^\vee}(\Delta_{w'})$
SetOfWalls
improved by internal
use of **Pool**
retrieve $\Omega(D')$
from $\bigcup_{i=1}^{N} \mathcal{R}_{k+1}^{(i)}$
set $\Omega(D')$ of walls
of the chamber D'
Database
update
▶ Store $\Omega(D')$ into $\mathcal{R}_{k+1}^{(j)}$
▶ Store w' into $\mathcal{E}_{k+1}^{(j)}$
$\Omega(D')$
K3SU.COM

Enforcing parallelism with this state of mind can be pushed a little further to bring an additional improvement to Borcherds' method. A fundamental principle of Borcherds' method is that the data tuple associated with a chamber D of level $k+2$ adjacent to a chamber in $\mathcal{L}_{k+1}$ along a non (-2)-wall is stored into $\mathcal{L}_{k+2}$ whenever D represents a new congruence class. Another basic rule of Borcherds' method is that the adjacencies of such chambers, i.e., the adjacencies of chambers discovered during the $(k+1)$-th iteration, are explored during the $(k+2)$-th iteration, and not earlier.

Viewing things in terms of parallel deployment enables us to bend this rule and think ahead. Indeed, delaying the exploration of the adjacencies of chambers adjacent to chambers discovered during the $(k+2)$-th iteration, i.e., the exploration of chambers of level $k+3$, until the next iteration no longer makes sense when parallelism can be enforced. We thus introduce an extra auxiliary process P_{N+1} tasked with the computation of the respective sets of walls of chambers of level $k+3$ adjacent to chambers in $\mathcal{L}_{k+2}$ along their non (-2)-walls, during the $(k+2)$-th iteration, and proceeding by the FIFO principle: First In, First Out. The process P_{N+1} is an instance of **APFB**. As soon as the primary process P_0 stores a chamber into $\mathcal{L}_{k+2}$, the process P_{N+1} explores its adjacencies along its non (-2)-walls and computes their respective sets of walls, following the mechanics of **APFB**. Sets of walls computed by P_{N+1} are stored into a set $\mathcal{R}_{k+1}^{N+1}$, while the Weyl vectors of the corresponding chambers are stored into a set $\mathcal{E}_{k+1}^{N+1}$. Both sets are part of the shared database and will be at the disposal of all processes during the $(k+2)$-th iteration. In terms of scalability, it is of course possible to assign additional processes to this task.

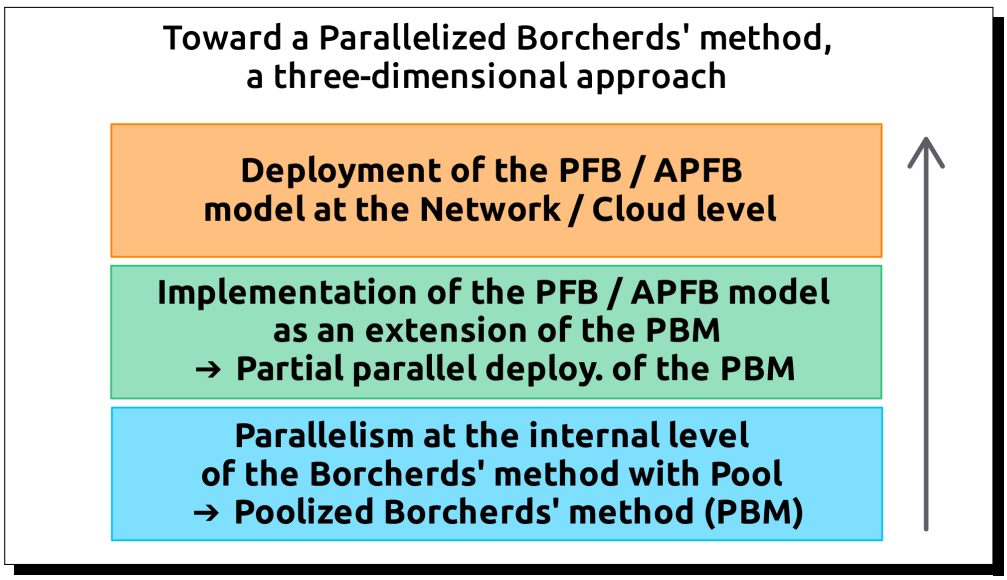
Our approach, in this thesis, toward a parallelized Borcherds' method can thus be summarized in the following figure:



The above structure is a not just a concept: It is fully operational, illustrated on concrete examples and explained on our website. The **PFB / APFB** strategy illustrated by the figure above can also be deployed at the network level to take advantage from the processing power of several machines.

There is no doubt that massive improvements can be brought to Borcherds' method by enforcing theoretical material related to the field of study of complex systems involving synchronization, parallelism, concurrency, sequencing and conflict management between processes.

The following diagram illustrates, in a concrete manner, our global approach toward a parallelized Borcherds' method. We did our best to provide a sound and safe starting point, a beachhead. We sincerely believe that everything remains to be done with regards to the deployment of Borcherds' method in parallel.



We kindly ask our readers to keep in mind that the initial goal of this thesis consisted in studying automorphism groups and orbits of (-2)-curves on $K3$ surfaces X_t with Picard number 3 for various values of the parameter $t \in \mathbb{Z}$. Ultimately, we provided computer-based solutions that enabled us to fulfill our end of the deal with full automation. These solutions have a much larger scope of application and thus opened many doors for further study. However, studying the parallel deployment of Borcherds' method was by no means the aim of this thesis. We did our best, with the tools at our disposal, and within the time constraints imposed by this doctoral project, to bring our ideas to life.
We, nevertheless, write it again: Everything remains to be done on the subject of parallelism & Borcherds' method and there is huge potential for development on the subject if this endeavor is carried out from an HPC perspective.

Parallelism and the Borcherds' method - Online content

Part II
A computer-based algorithmic approach to the study of projective models of $K3$ surfaces and unirationality of their moduli spaces

2 Projective models & unirationality

Smooth rational curves $C \simeq \mathbb{P}^1$ are central objects of study in the field of $K3$ surfaces. The term (-2)-curves is often used to refer to classes of smooth rational curves on $K3$ surfaces. Note that in this entire dissertation, we willingly make no distinction between a (-2)-curve and its class in the Néron-Severi group. In 2019, Pierre Lairez and Emre Can Sertöz published an article [9] in which can be found an algorithm to compute classes of smooth rational curves on $K3$ surfaces. This algorithm, which mobilizes material from Vinberg [21] and Shimada [18], inspired our advisor Professor Xavier Roulleau to produce an implementation which was then released along with the publication of his article [15] in 2019. Given the Gram matrix of the Néron-Severi group $\mathrm{NS}(X)$ of a $K3$ surface X, an integer m and an ample class $P_0 \in \mathrm{NS}(X)$, Roulleau's program **SmoothRationalCurves** outputs the set of classes $C \in \mathrm{NS}(X)$ of all smooth rational curves on X such that $C \cdot P_0 \leq m$. Roulleau designed his program in such a way that modifying a few lines of code and adding an input parameter d is enough to make his program capable of returning the set of all classes of curves D satisfying $D^2 = d$ and $D \cdot P_0 \leq m$. We have to mention how important this program was to us during during the early days of this thesis. Had this program never been produced by Roulleau, it is probable that our study would then have never been oriented toward the use of computer-based solutions for the study of $K3$ surfaces. We used Sage's Pythonic interface to Magma in order to integrate Roulleau's program into a Pythonic environment. We present the mechanics and the algorithmic structure behind this program in section 2.1. Our adaptation of this program is named **CGS**, for **C**lasses of any **G**iven **S**quare, and can be found under the same name in our **proj_mod** suite. We produced a real Python port of **CGS**, but this port did not bring any performance improvement over the version adapted from Roulleau's Magma program. The reason is that implementing this program requires short lattice vectors enumeration tools, a ground on which Magma (with ShortVectors, ShortVectorsProcess) crushed all the alternatives we had on hand during our thesis. The program **CGS** enabled us to start studying $K3$ surfaces by enforcing a computer-based algorithmic ap-

proach. For various positive integer values of an integer parameter $t > 0$, we were initially tasked with the study of projective models surfaces belonging to the family of $K3$ surfaces X_t with a Néron-Severi group having Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

with respect to a fixed basis. We put Roulleau's program to good use by combining it with existing results in order to uncover a wealth of information on these surfaces: Determination of projective models of these surfaces, criterion for the unirationality of their moduli spaces, computation a generating set of their automorphism group $\mathrm{Aut}(X_t)$ (see the Part 1 to this thesis, or click here), study of a fundamental domain of the action of $\mathrm{Aut}(X_t)$ onto $\mathrm{Nef}(X) \cap \mathcal{P}_S$, determination of explicit equation for these surfaces. In section 2.6, we build on a technique, used by Roulleau in his articles [16] and [15], which consists in taking advantage of the knowledge of a configuration of the form

$$\begin{cases} C_1 + C_2 = n_1 D \\ C_3 + C_4 = n_2 D \end{cases}$$

with $C_1, C_2, C_3, C_4 \in \mathrm{NS}(X)$ classes of smooth rational curves and D an ample class in order to study projective models of $K3$ surfaces and study the unirationality of their moduli spaces. Such configurations will be referred to as systems, see definition 42. Note that the procedure **CGS** presented in section 2.1 enables us to obtain concrete data regarding classes of smooth rational curves and divisors on $K3$ surfaces. For convenience, we produced a program to automatically form systems, as defined above, on a $K3$ surface. This program is called **SysFinder** and is available for download on K3surfaces.com.

A famous result that incorporates material from Saint-Donat [17] and Morrison [13], stated as Theorem 41 in this thesis, gives a precise characterization of the projective models that can be obtained from the morphism into projective space obtained associated with the complete the linear system $|D|$ of an ample class D of self-intersection $2, 4, 6$ or 8 and satisfying various prescribed conditions of base-point freeness and non-hyperellipticity. This approach presented two challenges that could only be overcome by producing new tools:

- How can we determine if a given class is ample or not, using a fully computer-based solution that can be deployed on any complex K3 surface? Our answer to this challenge is **AmpTester**.

- How can we deal with the base-point freeness and hyperellipticity conditions of the classical **SDM** Theorem so that we can escape the burden of handcrafting criteria for these notions specific to each surface under study? Our answer is **PModChecker**, for ***P**rojective **Mod**els **Checker***.

We thus ultimately produced tools that turned out to have a framework of application going way beyond the above-mentioned family of $K3$ surfaces. To deal with the first challenge, we produced a universal ampleness tester for classes of divisors on $K3$ surfaces, as explained in the section 2.2 of this thesis. As our thesis was nearing its end, we realized that we had all the algorithmic material in hand to give a full computer-based incarnation to Theorem 41 for the study of projective models. This classical and well-known theorem, widely known under its classical formulation involving the notions of based-point freeness and non-hyperellipticity, also possesses an equivalent formulation in terms of purely numerical conditions, that can be fully tested using a computer-based approach. Following this path, we took care of the second challenge. We give more details about this in the section 2.6, in which we illustrate all the solutions implemented during this portion of our thesis by applying them to the study of projective models and of the unirationality of moduli spaces of the $K3$ surfaces X_t with Néron-Severi group isomorphic to the integral lattice with Gram matrix $\mathrm{diag}(2t, -2, -2)$ with respect to a fixed basis. In section 2.4, we establish that

the discriminant group of these surfaces has no isotropic elements whenever the parameter t can be expressed as a product of distinct primes and satisfies $t \equiv 3 \pmod 4$. As shown in section 2.6, this result is useful when studying the unirationality of the moduli spaces of these surfaces. We have to mention that we took advantage of algorithmic material that can be found in Shimada's article [18] to deal with both challenges. Let us sum things up before going into the heart of the matter:

- In section 2.1, we introduce the mechanics behind the procedure **CGS**.
- In section 2.2, we present the inner workings of the universal ampleness tester, which requires as sole input the data of a Gram matrix of $\mathrm{NS}(X)$, of a known ample class, and of the class to be tested.
- In section 2.3, we explain how to exhibit an initial ample class on a given surface and provide a step-by-step example. Such an ample class can then be used as an ambient parameter for universal ampleness tester **AmpTester**.
- In section 2.4, we establish the result mentioned above on discriminant groups of surfaces X_t for cases where t is a product of distinct primes and satisfies $t \equiv 3 \pmod 4$.
- In section 2.5, we quickly review basic formulas on dimensions of linear systems of curves and hypersurfaces in projective spaces.
- We finally introduce **PModChecker** in section 2.6 and display how all these tools can be mobilized to determine projective models of $K3$ surfaces. We also put these tools to use to create concrete openings for the study of the unirationality of moduli spaces of the familiar $K3$ surfaces X_t with Néron-Severi group isomorphic to the integral lattice with Gram matrix $\mathrm{diag}(2t, -2, -2)$ with respect to a fixed basis.

2.1 Procedure CGS - Computing Classes of a Given Square

As indicated in the introduction to this thesis, the **SmoothRationalCurves** Magma program due to Roulleau had a substantial positive impact on our work in 2019 and was a key factor that helped us to put this thesis on track by adopting a computer-based approach to the study of $K3$ surfaces. Roulleau designed his program in such a way that only a few alterations in the code can widen its scope of application and turn this tool into a generator of data on classes of divisors D having a self-intersection $D^2 = d$ and satisfying $D \cdot P_0 \leq m$, where integers d, m, and an ample class P_0 are specified as input data by the user. The result of such an update is our program **CGS**, a generalized version of the initial program. As suggested by the name **SmoothRationalCurves**, Roulleau's initial Magma program, can identify classes of (-2)-curves among the elements of a set of (-2)-classes by enforcing an algorithm due to Vinberg [21]. The program **CGS** naturally inherits this feature when $d = -2$. We now present the algorithm behind the **SmoothRationalCurves** program. Note that the article [9] from Pierre Lairez and Emre Sertöz is authority content this matter. This subsection will hence be based on their material and formulated in the general case where the classes to be produced have self-intersection $d \geq -2$. Before proceeding further, let us get things straight about the notations used in this section:

- The capital letter S is used as a shorthand to denote the Néron-Severi group $\mathrm{NS}(X)$ of X.
- The greek letter ρ is used to denote the Picard number of X. That is, we set $\rho = \mathrm{rank}(S)$.
- We denote by $\mathcal{P}_S$ the positive cone of X, that is, $\mathcal{P}_S$ is the connected component of
$$\left\{D \in \mathrm{NS}(X) \otimes \mathbb{R} \mid D^2 > 0\right\}$$
containing ample classes.
- We denote by G_S a Gram matrix of S with respect to a fixed basis.

- We use the notation P_0 to denote a fixed ample class in $S = \mathrm{NS}(X)$.

Given the input data of G_S, of an ample class P_0 and of integers d and m, the following procedure due to Shimada [18] and Vinberg [21] outputs the list of classes $C \in \mathrm{NS}(X)$ such that

$$C^2 = d \qquad \text{and} \qquad C \cdot P_0 \leq m.$$

Procedure CGS: Assume that a basis for S is fixed. This basis will be referred to as the standard basis for S. We start by computing (e.g., by using a function from the SageMath library) a basis

$$\mathcal{B} = \{\lambda_1, \ldots, \lambda_{\rho-1}\}$$

of the rank $\rho - 1$ sublattice

$$\Lambda = P_0^{\perp} \cap (\langle P_0, P_0 \rangle_S \, S \oplus \mathbb{Z} P_0)$$

of $S \simeq \mathbb{Z}^3$ and then compute its Gram Matrix

$$G_\Lambda = M_{\mathcal{B}} G_S M_{\mathcal{B}}^T,$$

where $M_{\mathcal{B}}$ is the $((\rho - 1) \times \rho)$-sized matrix whose rows are taken to be the elements of $\mathcal{B}$. Since $\Lambda \subset P_0^{\perp}$, and since the ample class P_0 by definition satisfies

$$\langle P_0, P_0 \rangle_S > 0$$

the Hodge Index Theorem ensures that the restriction to Λ of the intersection form of S is negative definite. The strict inequality

$$-\langle D, D \rangle_\Lambda > 0$$

therefore holds for all $D \in \Gamma$. That is, the Gram Matrix G_Λ of Λ is negative definite. We then let

$$\alpha = \langle P_0, P_0 \rangle_S .$$

Using a short lattice vectors enumerator, we compute

$$\mathcal{L} = \left\{ D \in \Lambda \mid -\langle D, D \rangle_\Lambda \leq -d\alpha^2 + \alpha m^2 \right\} .$$

The enumerator will return elements of $\mathcal{L}$ as vectors with coordinates expressed in terms of the basis of Λ which has been computed earlier. This is however not a problem, since we have a basis $\mathcal{B}$ for Λ made of elements of S which enables us to express elements of $\mathcal{L}$ with respect to the standard basis of S. We assume that elements of $\mathcal{L}$ have thus been exprimed with respect to the standard basis of S. Let

$$\mathcal{A} = \left\{ -d\alpha^2 + \alpha y^2 \mid y \in [0 \ldots m] \right\}$$

To each $D \in \mathcal{L}$ is then associated the rational

$$m_D = \sqrt{\langle D, D \rangle_\Lambda + d\alpha^2} / \sqrt{\alpha}.$$

Define

$$\mathcal{L}' = \left\{ D \in \mathcal{L} \mid -\langle D, D \rangle_\Lambda \in \mathcal{A} \text{ and } \frac{1}{\alpha}(m_D P_0 + D) \in S \right\} .$$

and note that the condition

$$\frac{1}{\alpha}(m_D P_0 + D) \in S$$

holds for an element $D \in \mathcal{L}$ if and only if α divides each of the coordinates (w.r.t the basis of S) of $m_D P_0 + D$. To each element $D \in \mathcal{L}'$ can be associated an element $\Theta(D) \in S$ satisfying

$$\Theta(D) \cdot P_0 \leq m \qquad \text{and} \qquad \langle \Theta(D), \Theta(D) \rangle_S = d$$

where Θ is the transformation

$$\Theta : \mathcal{L}' \longrightarrow S$$

defined by

$$\Theta : D \longmapsto \frac{1}{\alpha}(m_D P_0 + D).$$

Indeed, we have:

$$\begin{aligned}
\langle \Theta(D), \Theta(D) \rangle_S &= \left\langle \frac{1}{\alpha}(m_D P_0 + D), \frac{1}{\alpha}(m_D P_0 + D) \right\rangle_S \\
&= \frac{1}{\alpha^2}(m_D^2 \langle P_0, P_0 \rangle_S + \langle D, D \rangle_S) \\
&= \frac{1}{\alpha^2}(- \langle D, D \rangle_S + d\alpha^2 + \langle D, D \rangle_S) \\
&= d
\end{aligned}$$

and

$$\begin{aligned}
\langle \Theta(D), P_0 \rangle_S &= \left\langle \frac{1}{\alpha}(m_D P_0 + D), P_0 \right\rangle_S \\
&= \frac{1}{\alpha} m_D \langle P_0, P_0 \rangle_S + \frac{1}{\alpha} \langle P_0, D \rangle_S \\
&= \frac{1}{\alpha} m_D \alpha + 0 \\
&= m_D
\end{aligned}$$

where we used the fact that $\langle P_0, D \rangle_S = 0$ because

$$D \in \mathcal{L} \subset P_0^{\perp} \cap (\alpha \mathrm{NS}(X) + \mathbb{Z} P_0).$$

Note that the assumption $D \in \mathcal{L}'$ implies that $0 \leq m_D \leq m$. We can thus compute the set

$$\mathcal{C}(m, d) = \left\{ \frac{1}{\alpha}(m_D P_0 + D) \mid D \in \mathcal{L}' \right\},$$

and this set is the desired set of classes of self-intersection d on X having intersection product with P_0 less than or equal to m. When

$$d \neq -2,$$

the procedure outputs $\mathcal{C}(m,d)$ and stops. Otherwise, $\mathcal{C}(m,-2)$ is the set of (-2)-classes having intersection product with P_0 less than or equal to m. Further processing is thus needed in order to identify classes of (-2)-curves among the (-2)-classes forming this set. Let

$$\mathcal{C}_m = \{D \in \mathrm{NS}(X) \mid \langle D, D\rangle_S = -2 \text{ and } \langle D, P_0\rangle_S = m\}$$

and define

$$\mathcal{N}_m = \{D \in \mathcal{C}_m \mid \forall p < m, \forall D' \in \mathcal{N}_p,\ \langle D, D'\rangle_S \geq 0\}\ .$$

Note that $\mathcal{N}_1 = \mathcal{C}_1$ holds, and that $\mathcal{N}_m$ can be computed recursively. Let $\mathcal{R}_m$ be the set of classes of smooth rational curves having intersection product with P_0 less than or equal to m. We show that there is a bijection between the sets $\mathcal{N}_m$ and $\mathcal{R}_m$. It is well-known that any two classes D, D' of irreducible curves satisfy

$$\langle D, D'\rangle_S \geq 0.$$

Thus, if $D \in \mathcal{R}_m$, then $D \in \mathcal{N}_m$. The set $\mathcal{R}_m$ is therefore a subset of the set $\mathcal{N}_m$, i.e., we have

$$\mathcal{R}_m \subseteq \mathcal{N}_m. \tag{2.1}$$

Let $C \in \mathcal{N}_m$. The Riemann-Roch theorem for surfaces gives that one of the strict inequalities

$$\dim\ H^0(X, \mathcal{O}_X(C)) > 0 \qquad \text{or} \qquad \dim\ H^0(X, \mathcal{O}_X(-C)) > 0$$

must hold. That is, either C or $-C$ is the class of an effective curve. Since $C \in \mathcal{N}_m$, we have

$$\begin{aligned} \langle C, P_0 \rangle_S &= m \\ &> 0 \end{aligned}$$

and deduce that C must be the class of an effective curve, because otherwise the intersection product of C with P_0 would not be positive. Using the fact that the class of an effective curve can we written as the sum of classes of distinct irreducible curves, we can express C as

$$C = \sum_i \beta_i C_i$$

where all the coefficients satisfy $\beta_i > 0$ and the classes in this formal sum are such that

$$\langle C_i, C_j \rangle_S \geq 0$$

whenever $i \neq j$. Since the class C has self-intersection -2, there exists an integer k such that the strict inequality

$$\langle C, C_k \rangle_S < 0.$$

holds. The adjunction formula then ensures that C_k satisfies

$$C_k^2 = -2.$$

The class C_k being of self-intersection -2 and irreducible is therefore the class of a smooth rational curve on X. Let

$$m' = \langle C_k, P_0 \rangle_S .$$

Since, by assumption, we have $\langle C, P_0 \rangle_S = m$, it is clear that

$$0 < m' \leq m$$

holds. Since the set $\mathcal{R}_{m'}$ is a subset of $\mathcal{N}_{m'}$, we moreover have

$$C_k \in \mathcal{N}_{m'}.$$

However, the fact that

$$\langle C_k, C \rangle_S < 0$$

contradicts the definition of $\mathcal{N}_m$. Hence

$$m' = m \qquad \text{and} \qquad C_k = C$$

This pattern of proof by contradiction enables us to assert that each element of $\mathcal{N}_m$ is in fact the class of a smooth rational curve, hence contained in $\mathcal{R}_m$.

Thus, we have

$$\mathcal{N}_m \subseteq \mathcal{R}_m$$

and deduce from (2.1) that the equality

$$\mathcal{N}_m = \mathcal{R}_m$$

holds. The fact that $\mathcal{N}_m$ is computable recursively thus provides a mean to identify classes of (-2)-curves among sets of (-2)-classes. Roulleau's Magma program **SmoothRationalCurves** relies on these mechanics.

2.2 Universal Ampleness Tester

Let X be $K3$ surface over $\mathbb{C}$ with Néron-Severi group $S = \mathrm{NS}(X)$. When working with surfaces on which lie a finite number of smooth rational curves, determining whether a class is ample or not is a non-issue. Indeed, it is well-known that a class $D \in \mathrm{NS}(X)$ is ample if and only if it satisfies

$$\langle D, C \rangle_S > 0$$

for all classes D of smooth rational curves on X. Only a finite number of intersection products have therefore to be computed in order to get an answer on the ampleness of a class D on a $K3$ surface on which lie a finite number of smooth rational curves, i.e., having a finite automorphism group. However, this approach is pointless when the $K3$ surface under study has an infinite number of smooth rational curves lying on it, that is, on surfaces for which $\mathrm{Aut}(X)$ is infinite. For such surfaces, lifting the veil on the ampleness or non-ampleness of classes has always been a problem until now. Our solution to this issue is based on the fact that Shimada fortunately devoted eight lines of his 2013 article [19, p.31/32] to outline a characterization of ampleness which led us to produce a universal ampleness tester capable of testing whether any class $D \in S$ is ample or not provided that we have prior knowledge of one ample class. We thus smashed the door slightly opened by Shimada's almost a decade ago and gave life to a universal ampleness tester: **AmpTester**. Note that starting from this line, we stop using capital letters to denote classes in S and do so for the sake of clarity until the end of this section. Assume known an ample class $a_0 \in S$. Shimada states that a class $v \in S$ is ample if and only if the three following conditions are satisfied:

- **Condition AC1:** Both inequalities

  $$\langle v, v \rangle_S > 0 \quad \text{and} \quad \langle v, a_0 \rangle > 0$$

  hold, so that $v \in \mathcal{P}_S$.

- **Condition AC2:** The set

$$\{r \in S \mid \langle v, r\rangle_S = 0,\ \langle r, r\rangle_S = -2\}$$

  is empty.

- **Condition AC3:** The set

$$\{r \in S \mid \langle v, r\rangle_S < 0,\ \langle a_0, r\rangle_S > 0,\ \langle r, r\rangle_S = -2\}$$

  is empty. That is, the line segment in $\mathcal{P}_S$ connecting a_0 and v does not intersect any hyperplane $(r)^\perp$ perpendicular to some (-2)-class $r \in S$.

Checking whether condition **AC1** holds is not a problem. Things are not as simple regarding conditions **AC2** and **AC3**. In his article [18], Shimada fortunately provides algorithms that can be used to compute the sets involved in verifying these conditions.

- Algorithms 3.1 and 3.2 from [18] can be used to check whether **AC2** holds. We already know Shimada's Algorithm 3.1 as **ShiVectors**, described in the section 1.4 of this thesis. We gave the name **ShiChecker** to our implementation of Shimada's Algorithm 3.2 and explain how to implement it in section 2.2.1.

- Algorithms 3.1 and 3.3 from [18] can be used in order to check whether **AC3** holds. Let us give the name **ShiBooster** to algorithm 3.3. We explain how to deal with its implementation in section 2.2.2.

We took on the challenge and gave life to Shimada's idea of a universal ampleness tester. The result is **AmpTester**, detailed and available on K3surfaces.com We also combined Shimada's idea with Roulleau's program **SmoothRationalCurves** in order to make the program **AmpTester** capable of returning classes C of smooth rational curves such that $D \cdot C \leq 0$ whenever D is not ample and has positive self-intersection, thus providing an additional and concrete evidence of the non-ampleness of D thus supporting the findings of **AmpTester**.

2.2.1 ShiChecker - Checking AC2

Let L be a lattice and $u \in L$ such that

$$\langle u, u \rangle_L > 0$$

Let α and δ be integers. We now explain how we implemented the algorithm to compute sets of the form

$$\mathcal{H} = \{x \in L \mid \langle x, u \rangle_L = \alpha,\ \langle x, x \rangle_L = \delta\}$$

outlined by Shimada in his article [18, Algorithm 3.2].

Our implementation of this algorithm is called **ShiChecker** and is available for download on K3surfaces.com. The general strategy to do so is based on the fact that an element $v \in \mathcal{H}$ can be obtained by

(i) determining a solution $c \in S$ of the equation $\langle x, u \rangle_S = \alpha$ and then

(ii) determining an element $y \in u^\perp \subset S$ satisfying

$$\langle y + c, y + c \rangle_S = \delta,$$

that is, satisfying

$$\langle y, y \rangle_S + 2\langle y, c \rangle_S + \langle c, c \rangle_S = \delta. \tag{2.2}$$

The data of c and y can then be used to assemble an element

$$v = y + c$$

which will then satisfy

$$\begin{aligned}\langle v, u\rangle_S &= \langle y + c, u\rangle_S \\ &= 0 + \langle c, u\rangle_S \\ &= \alpha\end{aligned}$$

and

$$\begin{aligned}\langle v, v\rangle_S &= \langle y + c, y + c\rangle_S \\ &= \delta,\end{aligned}$$

so that $v \in \mathcal{H}$, as desired.

Implementation of *(i)*: An element $x \in S$ can be represented by a coordinate vector

$$x = [x_1, \ldots, x_\rho]_S$$

where $\rho = \mathrm{rank}(S)$. Solving the equation

$$\langle x, u\rangle_{S^\vee} = \alpha$$

for $x \in S$ amounts to determining integers $x_1, \ldots, x_\rho$ such that

$$\begin{bmatrix} x_1 & \ldots & x_\rho \end{bmatrix} G_S \begin{bmatrix} u_1 \\ \vdots \\ u_\rho \end{bmatrix} = \alpha. \tag{2.3}$$

The left-hand side of this expression can be expanded an re-arranged so that equality (2.3) can be written as an expression of the form

$$\sum_{i=1}^{\rho} \gamma_i x_i = \alpha$$

where the γ_i are elements of $\mathbb{Q}$. Clearing the eventual denominators on both sides of this expression yields an equality of the form

$$\sum_{i=1}^{\rho} \mu_i x_i - \gamma = 0 \tag{2.4}$$

where $\gamma \in \mathbb{Z}$ and $\mu_i \in \mathbb{Z}$ for $1 \leq i \leq \rho$. The resolution of this degree 1 multivariate equation is then accomplished using a CAS such as Sage and gives us a basis $\{\epsilon_1, \ldots, \epsilon_{\rho-1}\}$ of the $(\rho-1)$-dimensional solution space of this equation. That is, solutions of (2.4) are generated by

$$\epsilon(t_0, \ldots, t_{\rho-1}) = \epsilon_1 t_1 + \cdots + \epsilon_{\rho-1} t_{\rho-1} \tag{2.5}$$

where $t_i \in \mathbb{Z}$ for $1 \leq i \leq \rho - 1$.

Implementation of *(ii)*: We have seen how to generate solutions of the equation $\langle x, u \rangle_S = \alpha$. Assume such a solution $c \in S$ fixed. We now explain Shimada's procedure to obtain an element $y \in u^{\perp} \subset S$ satisfying

$$\langle y + c, y + c \rangle_S = \delta.$$

Since a Gram matrix of $S = \mathrm{NS}(X)$ is by design indefinite, we cannot use a short lattice vectors enumeration algorithm to determine elements $x \in S$ satisfying $\langle x, x \rangle_S = \delta$. Shimada's idea to overcome this obstacle consists in finding a particular sublattice of S on which the restriction of the bilinear form is definite. We have $\langle u, u \rangle_S > 0$ by assumption, hence the Hodge Index theorem gives us that the restriction of $\langle\ ,\ \rangle_{S^\vee}$ to the orthogonal complement $u^{\perp}$ of u in S is negative definite. Recall that we explained in section 1.4 how to proceed to implement Shimada's short lattice vectors enumeration algorithm from his article [18, Section 3.1] to determine solutions of expressions of the form.

$$xQx^t + 2xL + c \leq 0$$

This tool then enables us to determine elements $y \in u^{\perp} \subset S$ satisfying the equality

$$\langle y, y\rangle_S + 2\langle y, c\rangle_S + \langle c, c\rangle_S = \delta.$$

Since S is an integral lattice, it is contained in its dual, i.e., $S \subset S^{\vee}$. By definition, an element $x \in S$ belongs to $u^{\perp}$ if and only if $\langle x, u\rangle_S = 0$. Solving this equation amounts to determining integers $x_1, \ldots, x_\rho$ such that

$$\begin{bmatrix} x_1 & \ldots & x_\rho \end{bmatrix} G_S \begin{bmatrix} u_1 \\ \vdots \\ u_\rho \end{bmatrix} = 0 \tag{2.6}$$

Expanding and clearing the denominators, we obtain from the above equality a first-degree multivariate linear equation of the form

$$\sum_{i=1}^{\rho} \gamma_i x_i = 0.$$

which can easily be solved for integral solutions using a CAS. We thus obtain a basis $\{\xi_1, \ldots, \xi_{\rho-1}\}$ of the solution space of this equation, so that its solutions can be generated using

$$\xi(t_1, \ldots, t_{\rho-1}) = \xi_1 t_1 + \cdots + \xi_{\rho-1} t_{\rho-1},$$

where the t_i are integers for $0 \leq i \leq \rho - 1$. Using the basis of $u^{\perp}$ that we now have at our disposal enables us to compute a Gram Matrix $G_{u^{\perp}}$, which is negative definite. That is, we compute the matrix with entries $\langle \xi_i, \xi_j\rangle_S$ for $1 \leq i, j \leq \rho - 1$. Let $p_\alpha \in S$ denote a fixed solution of the equation $\langle x, u\rangle_S = \alpha$, whose resolution was explained earlier when we dealt with point *(i)*. We now determine an element $y \in u^{\perp} \subset S$ such that

$$\langle y + p_\alpha, y + p_\alpha\rangle_S = \delta. \tag{2.7}$$

As mentioned previously, the element $v = y + p_\alpha$ will then satisfy

$$\langle v, v\rangle_S = \delta \qquad \text{and} \qquad \langle v, u\rangle_S = \alpha$$

so that we will have $v \in \mathcal{H}$. We have seen in section 1.4 that Shimada's Short lattice vectors custom algorithm **ShiVectors** takes a positive quadratic triple

$$[Q, P, c]$$

as input data, where

- Q is a $n \times n$-sized symmetric positive definite integral matrix,
- P is a $(1 \times n)$-sized column vector with integer entries,
- c is a rational parameter.

We recall that Shimada ensures that the procedure **ShiVectors** outputs the finite set

$$\{x \in \mathbb{Z}^n \mid q_{QT}(x) \leq 0\}$$

of solutions of

$$xQx^t + 2xP + c \leq 0$$

Let us arrange (2.7) to make it comply with this format. We first replace the equality sign in

$$\langle y + p_\alpha, y + p_\alpha\rangle_S = -2 - \beta \tag{2.8}$$

by an $\leq$ sign, and note that there is no loss of generality in doing so. We expand and arrange (2.8) to obtain:

$$\langle y, y\rangle_S + 2\langle y, p_\alpha\rangle_S + \langle p_\alpha, p_\alpha\rangle_S - \delta \leq 0. \tag{2.9}$$

Recall that we obtained a basis

$$\{\xi_1, \ldots, \xi_{\rho-1}\}$$

for $u^{\perp}$ earlier. Since the element y is assumed to belong to $u^{\perp}$, any short lattice vectors enumerator executed using the Gram matrix of $u^{\perp}$ will return elements having coordinates given with respect to the basis of $u^{\perp}$ from which the Gram matrix was obtained, which, in our case, is the above-mentioned basis. Denote by $y_1, \ldots, y_{\rho-1}$ the coordinates of y with respect to the latter. That is,

$$y = y_1\xi_1 + \cdots + y_{\rho-1}\xi_{\rho-1}.$$

The term $2\langle y, p_\alpha\rangle_{S^\vee}$ in expression (2.9) can then be re-arranged as follows:

$$\begin{aligned}
2\langle y, p_\alpha\rangle_S &= 2\langle y_1\xi_1 + \cdots + y_{\rho-1}\xi_{\rho-1}, p_\alpha\rangle_S \\
&= 2(y_1\langle \xi_1, p_\alpha\rangle_S + \cdots + y_{\rho-1}\langle \xi_{\rho-1}, p_\alpha\rangle_S) \\
&= 2\begin{bmatrix} y_1 & \ldots & y_{\rho-1}\end{bmatrix}\begin{bmatrix} \langle \xi_1, p_\alpha\rangle_S \\ \vdots \\ \langle \xi_{\rho-1}, p_\alpha\rangle_S \end{bmatrix} \\
&= 2yP
\end{aligned}$$

where

$$P = \begin{bmatrix} \langle \xi_1, p_\alpha\rangle_S \\ \vdots \\ \langle \xi_{\rho-1}, p_\alpha\rangle_S \end{bmatrix}.$$

The inequality

$$\langle y + p_\alpha, y + p_\alpha\rangle_S \leq -2 - \beta$$

can therefore be written as

$$y\, G_{u^\perp}\, y^t + 2yP + c \leq 0$$

where

$$y = [y_1, \ldots, y_{\rho-1}] \qquad \text{and} \qquad c = \langle p_\alpha, p_\alpha\rangle_S - \delta.$$

Again, we recall that the input data format for Shimada's algorithm **ShiVec-**

tors algorithm consists in a positive quadratic triple $[Q, L, c]$ used to define a quadratic function of the form

$$yQy^t + 2yL + c, \tag{2.10}$$

where Q is required to be a positive definite matrix. Since the Gram matrix $G_{u^\perp}$ of $u^\perp$ is negative definite (Hodge Index Theorem), we will use $-G_{u^\perp}$, which is positive definite, as input for the short lattice vectors algorithm **ShiVectors** instead of $G_{u^\perp}$. Anyways, taking the negative of expression 2.10 with $Q = G_{u^\perp}$ gives

$$y(-G_{u^\perp})y^t + 2y(-L) + (-c)$$

and we thus obtain that the triple to be used as input data for Shimada's algorithm **ShiVectors** is

$$[-G_{u^\perp}, -L, -c] = \left[-G_{u^\perp}, -\begin{bmatrix} \langle \xi_1, p_\alpha \rangle_S \\ \vdots \\ \langle \xi_{\rho-1}, p_\alpha \rangle_S \end{bmatrix}, -\langle p_\alpha, p_\alpha \rangle_S + \delta\right].$$

This algorithm provides the data of elements $q \in u^\perp$ satisfying

$$\langle q + p_\alpha, q + p_\alpha \rangle_S \leq \delta,$$

from which we can readily obtain the elements $q \in u^\perp$ satisfying the equality

$$\langle q + p_\alpha, q + p_\alpha \rangle_S = \delta.$$

Let $v = q + p_\alpha$. It is clear that we have $\langle v, v \rangle_S = \delta$. The fact that $q \in u^\perp$ gives us that $\langle q, u \rangle_S = 0$. Since p_α is assumed to be a solution of $\langle x, u \rangle_S = \alpha$, we have $\langle p_\alpha, u \rangle_S = \alpha$. Hence

$$\begin{aligned} \langle v, u \rangle_S &= \langle q + p_\alpha, u \rangle_S \\ &= 0 + \langle p_\alpha, u \rangle_S = \alpha. \end{aligned}$$

We thus see that this procedure indeed enables us to obtain elements of the set

$$\mathcal{H} = \{x \in L \mid \langle x, u\rangle_L = \alpha,\ \langle x, x\rangle_L = \delta\}\,.$$

2.2.2 ShiBooster - Checking AC3

Assume that vectors $v, h \in S$ satisfying

$$\langle v, v\rangle_S > 0, \quad \langle h, h\rangle_S > 0 \quad \text{and} \quad \langle h, v\rangle_S > 0$$

are given. In his article [18, Section 3.3], Shimada describes an algorithm to compute the set

$$\mathcal{F} = \{r \in S \mid \langle r, h\rangle_S > 0, \langle r, v\rangle_S < 0, \langle r, r\rangle_S = d\}\,.$$

Our implementation of Shimada's algorithm is named **ShiBooster**. Note that it is available for download on K3surfaces.com. We follow Shimada's guidelines available in his article [18]. We start by computing the orthogonal complement

$$W = (h)^{\perp}$$

in S of the element $h \in S$ which is assumed to be given. We then define a projection

$$\mathrm{pr}_W : S \otimes \mathbb{Q} \longmapsto W \otimes \mathbb{Q}$$

sending an element $b \in S \otimes \mathbb{Q}$ to its projection $\mathrm{pr}_W(b)$ onto $W \otimes \mathbb{Q}$. For convenience, we will work in the framework of the duals $S^\vee$ and $W^\vee$ of the lattices S and W until the end of this subsection. Let

$$x = [x_1, \ldots, x_{\rho-1}]$$

be a $(\rho - 1)$-sized row vector made of formal variables x_i for $1 \leq i \leq \rho - 1$.

Consider the negative inhomogenous quadratic function

$$f : W \to \mathbb{Q}$$

defined by

$$f(x) : x \longmapsto \langle x, x \rangle_W + \frac{\langle h, h \rangle_S}{\langle v, h \rangle_S^2} \langle x, \mathrm{pr}_W(h) \rangle_W^2 .$$

We then formally expand the expression on the right-hand side, collect the terms, and form a negative definite matrix

$$M_f = [a_{ij}]$$

where a_{ij} is the coefficient of the term $x_i x_j$, $1 \leq i,\ j \leq \rho - 1$, in the expanded expression of f. The matrix $-M_f$ is positive definite, and we let L_f be the lattice with Gram Matrix $-M_f$. Using a short lattice vectors enumerator, we compute the set

$$\mathscr{S} = \left\{ b \in L_f \mid \langle b, b \rangle_{L_f} \leq 2 \right\}$$

Due to the fact that M_f has been obtained by taking the coefficients of f, the set $\mathscr{S}$ coincides with the set

$$\{ b \in W \mid f(b) \geq -2 \} .$$

We associate the quantity

$$\eta_b = \frac{-2 - \langle b, b \rangle_W}{\langle h, h \rangle_S}$$

to each element $b \in \mathscr{S}$, where we note that

$$-2 - \langle b, b \rangle_W > 0$$

holds since $b \in \mathscr{S}$, and that $\langle h, h \rangle_S > 0$ holds by assumption. Denote by $M_{W^\vee}$

the matrix formed by taking as row vectors the basis elements

$$\omega_1^\vee, \ldots, \omega_{\rho-1}^\vee \in S \otimes \mathbb{R}$$

of $W^\vee$, which can quickly be obtained with a computer. Assuming that an element $b \in \mathscr{S}$ is represented as a $(\rho - 1)$-sized column vector containing its coordinates with respect to the basis of $W^\vee$ mentioned above, we send an element $b \in W^\vee$ to an element $b^{S\otimes\mathbb{R}} \in S \otimes \mathbb{R}$ by the map

$$b \longmapsto bM_{W^\vee}.$$

Define an initially empty set $\mathcal{F} = \{\,\}$. For each $b \in \mathscr{S}$, we define

$$b^* = \sqrt{\eta_b}\, h + b^{S\otimes\mathbb{R}}.$$

If b^* satisfies the three following conditions

$$b^* \in S, \qquad \langle b^*, h\rangle_S > 0, \qquad \langle b^*, v\rangle_S < 0,$$

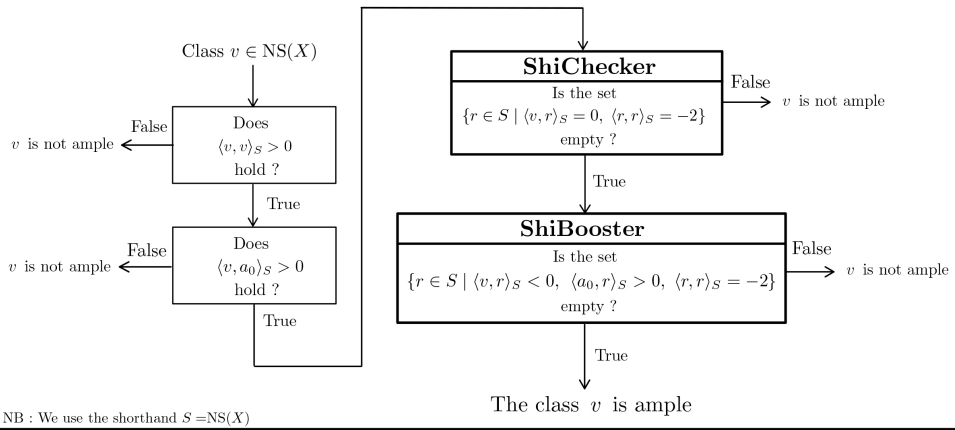
then we append b^* to $\mathcal{F}$. When all $b \in \mathcal{S}$ have thus been tested, Shimada ensures that the set $\mathcal{F}$ coincides with the desired set

$$\left\{r \in S \mid \langle r, h\rangle_S > 0, \langle r, v\rangle_L < 0, \langle r, r\rangle_L = d\right\}.$$

During ampleness testing, the initial ample class plays the role of h while class whose ampleness is to be determined plays v. Combining the programs **ShiBooster** and **ShiChecker**, we obtain our universal ampleness tester for classes of divisors on $K3$ surfaces, described in a figure on the following page.

AmpTester

⚠ Prior knowledge of one (and only one) ample class $a_0 \in \mathrm{NS}(X)$ is required in order to use **AmpTester** ⚠



NB : We use the shorthand $S = \mathrm{NS}(X)$

2.3 Finding an initial ample class

Having prior knowledge of an ample class is a prerequisite to executing many of the procedures encountered in this thesis. For example, an initial ample class is needed to test whether the initial chamber used in Borcherds' method is nondegenerate, an initial ample class is required to use the universal ampleness tester. Therefore, it is a matter of decency that we provide guidelines to determine an initial ample class. Assume given a complex $K3$ surface X with Néron-Severi group $S = \mathrm{NS}(X)$ and assume that we have no prior knowledge of any ample class. Given a class $v \in S$ satisfying

$$\langle v, v \rangle_S > 0,$$

a classical result that can be found in Huybrechts' book [5] states that there exists a transformation ω in the Weyl group of X such that $\pm\omega(v)$ is ample whenever the set

$$\{r \in S \mid \langle v, r \rangle_S = 0,\ \langle r, r \rangle = -2\}$$

is empty. In this case, the class $v \in S$ can thus be viewed as ample up to transformations in the Weyl group. We show how this strategy can be executed on a concrete example. Assume that the $K3$ under study is a surface X_t with Néron-Severi group $S_t = \mathrm{NS}(X_t)$ having Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

with respect to some fixed basis, and that the integer parameter t satisfies $t > 1$. Let us show that the class

$$P_0 = [2, -1, -1]\,.$$

can be taken as ample in $\mathrm{NS}(X_t)$ for all $t > 1$. We start by checking whether this class has strictly positive self-intersection. We have

$$\langle P_0, P_0 \rangle_{S_t} = 8t - 4$$

which is a strictly positive quantity when $t > 1$. Let us show that the set

$$\left\{ C \in S_t \mid \langle P_0, C \rangle_{S_t} = 0,\ \langle C, C \rangle_{S_t} = -2 \right\}$$

is empty whenever $t > 1$. Before proceeding further, note that what comes next can be done in a matter of seconds using a computer. We, however, proceed by hand for the sake of completeness of this thesis. Let us compute a basis of

$$(P_0)^{\perp} = \left\{ C \in S_t \mid \langle P_0, C \rangle_{S_t} = 0 \right\}$$

and then show that elements $C \in (P_0)^{\perp}$ of self-intersection -2 cannot exist. In order to compute a basis for $(P_0)^{\perp}_{S_t}$ we fix a class

$$D = [x, y, z]$$

in S_t with x, y, z integers not all equal to zero and assume that $D \in (P_0)^{\perp}$. From

$$\langle P_0, D \rangle_{S_t} = 0$$

we readily obtain that

$$z = -tx - y$$

so that D can be expressed as

$$\begin{aligned} D &= [x, y, -tx - y]_{S_t} \\ &= x\,[1, 0, -t] + y\,[0, 1, -1] \\ &= xB_{1,t} + yB_2 \end{aligned}$$

where

$$B_{1,t} = [1, 0, -t] \qquad \text{and} \qquad B_2 = [0, 1, -1]\,.$$

The orthogonal complement $(P_0)^\perp$ can thus be viewed as a sublattice of S_t spanned by the elements $B_{1,t}$ and B_2 of S_t. A Gram matrix

$$\begin{pmatrix} 2t(1-t) & -2t \\ -2t & -4 \end{pmatrix}$$

of this sublattice is then computed. Assume than an element $C = [u, v]$, with $u, v \in \mathbb{Z}$ not both equal to zero, belongs to $(P_0)^\perp$ and has self-intersection -2. Using the Gram matrix of $(P_0)^\perp$ to compute this self-intersection, we see that this assumption is equivalent to

$$tu^2(t-1) + 2tuv + 2v^2 = 1. \tag{2.11}$$

Note that the right-hand side of this equality is odd. Two possibilities arise regarding the left-hand side of this expression:

- Assume that $t = 2k \pm 1$ for some $k \in \mathbb{Z}$, that is, assume that t is an odd integer. Then $t - 1$ is even so that $tu^2(t-1)$ is also even.
- Assume that t is even. Then $tu^2(t-1)$ is even.

No matter the value of $t > 1$, the left-hand side of the equality (2.11) is therefore even, as a sum of even quantities. The left-hand side of (2.11) being odd, we see that assuming the existence of a non-trivial element in $(P_0)^\perp$ having self-intersection -2 leads to a contradiction. We therefore deduce that the set

$$\left\{C \in S_t \mid \langle P_0, C\rangle_{S_t} = 0,\ \langle C, C\rangle_{S_t} = -2\right\}$$

is empty. The result mentioned at the beginning of this section then enables us to consider the class $P_0 = [2, -1, -1]$ as ample in S_t for all $t > 1$, up to transformations in the Weyl group of X. When $t = 1$, proceeding analogously yields that $P_0 = [1, -1, -1]$ can be taken as ample.

For all $t \geq 1$, the data of an ample class P_0 enables us to enforce **AmpTester** to test any class in $\mathrm{NS}(X)$ for ampleness. Practical applications of our program **AmpTester** are extensively detailed on K3surfaces.com.

```
sage: amp0 = Matrix([2,-1,-1])
sage: AmpTester(Matrix([122,-2,-1]))

    ⓘ The known ample class amp0 is [ 2 -1 -1]

    ⓘ The K3 surface under study has Néron-Severi lattice with Gram matrix :

[14  0  0]
[ 0 -2  0]
[ 0  0 -2]

********************************************************************************

    ✓ The class [122  -2  -1] is ample !

    ✓ AmpTester Boolean value is : True

True
sage:
```

```
sage: AmpTester(Matrix([2000,51,355,55,999]))

    ⓘ The known ample class amp0 is [ 2 -1 -1 -1 -1]

    ⓘ The K3 surface under study has Néron-Severi lattice with Gram matrix :

[42  0  0  0  0]
[ 0 -2  0  0  0]
[ 0  0 -2  0  0]
[ 0  0  0 -2  0]
[ 0  0  0  0 -2]

********************************************************************************

    ⚠ The class [2000   51  355   55  999] is NOT AMPLE !

    ⚠ Note that the (-2)-curves contained in

[[0 0 0 0 1], [0 0 1 0 0], [0 0 0 1 0], [0 1 0 0 0]]

       are obstructions to the ampleness of [2000   51  355   55  999] !

    ⚠ Indeed, their intersection product with the class [2000   51  355   55  999] is strictly negative !

    ⚠ AmpTester Boolean value is : False
```

2.4 A useful result on the discriminant group of $\mathrm{NS}(X_t)$

A result from Curtis T. McMullen's article [11] states that given an even lattice, there is a one-to-one correspondence between the set of its overlattices and the set of subgroups of its discriminant group on which the restriction of the associated quadratic form vanishes. Assume that L is an even lattice having the property that its discriminant group $L^\vee / L$ has no non-trivial isotropic elements. Any element $x_0 \in L^\vee / L$ satisfying

$$q_L(x_0) = 0,$$

is then necessarily the identity element of $L^\vee / L$, i.e., $x_0 \in L$. In this case, the result mentioned above enables us to assert that L has no proper overlattices. This result will be key to us in order to exhibit $K3$ surfaces X_t for which the unirationality of the moduli space can be asserted: We enforce a technique due to Roulleau in [15] and show that under special conditions a quartic surface $\mathcal{Q}$ such that

$$\mathrm{NS}(X_t) \subseteq \mathrm{NS}(\mathcal{Q})$$

can be built from scratch using projective parameters. These conditions, when fulfilled, enable us to assert that the discriminant group of $\mathrm{NS}(X_t)$ has no non-trivial isotropic elements. As we just discussed, it is then be possible to assert that $\mathrm{NS}(X_t)$ has no proper overlattices so that the above inclusion becomes

$$\mathrm{NS}(X_t) \simeq \mathrm{NS}(\mathcal{Q})$$

hence establishing the unirationality of the moduli space of $K3$ surfaces with Néron-Severi group isomorphism to $\mathrm{NS}(X_t)$. Combining basing arithmetic and advanced computer-based algorithmic solutions, we will provide examples for which such a situation occurs. Our first objective consists in determining conditions under which the discriminant group of $\mathrm{NS}(X_t)$ has no isotropic element.

We established the following result which enables us to assert that $\mathrm{NS}(X_t)$ has no strict overlattices for infinitely many values of the parameter t :

Proposition 38. *If t is a product of distinct primes satisfying $t \equiv 3 \pmod 4$, then discriminant group $S_t^\vee / S_t$ of the lattice $S_t = \mathrm{NS}(X_t)$ has no isotropic element.*

We now provide a rigorous proof of this result. Before proceeding, recall that S_t is a shorthand for $\mathrm{NS}(X_t)$ and that a Gram matrix with respect to some fixed basis for the latter is assumed to be equal to

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}.$$

The diagonal shape of this matrix enables us to immediately state the following quite obvious result:

Proposition 39. *There is an isomorphism*

$$S_t^\vee / S_t \simeq (\mathbb{Z}/2t\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z}) .$$

Before proceeding further, let us see how things work regarding elements of the discriminant group of S_t. Let $t > 2$ be an integer, and assume that it can be expressed a product of distinct primes. We use the classical coordinate vectors notation to represent elements of $x \in S_t$ as

$$\begin{aligned} x &= x_1 v_1 + x_2 v_2 + x_3 v_3 \\ &= [x_1, x_2, x_3]_S \end{aligned}$$

where the elements

$$v_1 = [1,0,0]_S\,, \qquad v_2 = [0,1,0]_S \qquad \text{and} \qquad v_3 = [0,0,1]_S$$

are assumed to form a basis for S_t with the above-mentioned Gram matrix.

Applying the definition of the dual of a lattice which states that $S_t^\vee$ is formally defined as

$$S_t^\vee = \{x \in S \otimes \mathbb{Q} \mid \forall y \in S_t,\ \langle x, y\rangle_{S_t} \in \mathbb{Z}\}\,.$$

we see that an element $x \in S_t \otimes \mathbb{Q}$ expressed as $[x_1, x_2, x_3]_S$ satisfies $x \in S_t^\vee$ if and only if

$$\langle x, y\rangle_{S_t} \in \mathbb{Z}$$

holds for all $y \in S_t$. That is, if and only if

$$\langle x, v_1\rangle_{S_t} = 2tx_1 \in \mathbb{Z}, \quad \langle x, v_2\rangle_{S_t} = -2x_2 \in \mathbb{Z} \ \text{ and } \ \langle x, v_3\rangle_{S_t} = -2x_3 \in \mathbb{Z}.$$

That is, there exist integers a, b and c such that

$$x_1 = \frac{a}{2t}, \qquad x_2 = -\frac{b}{2} \qquad \text{and} \qquad x_3 = -\frac{c}{2}.$$

The quotient $S_t^\vee / S_t$ can thus be expressed as

$$S_t^\vee / S_t = \left\{ \left(\frac{a}{2t}, -\frac{b}{2}, -\frac{c}{2} \right) \mid a, b, c \in \mathbb{Z} \right\} / \left(\mathbb{Z}\,[1,0,0]_S + [0,1,0]_S + \mathbb{Z}\,[0,0,1]_S\right).$$

We use the notation $\overline{w}$ to denote the class in $S_t^\vee / S_t$ of an element

$$w = \left(\frac{a}{2t}, -\frac{b}{2}, -\frac{c}{2} \right) \in S_t^\vee.$$

Since S_t is an even lattice, the $\mathbb{Z}$-valued symmetric bilinear form on S_t extends to a $\mathbb{Q}$-valued symmetric bilinear form on $S_t^\vee$. The latter in turns defines a quadratic form

$$q : S_t^\vee / S_t \longrightarrow \mathbb{Q}/2\mathbb{Z}$$

defined by

$$q : \overline{x} \longmapsto x^2 \bmod 2\mathbb{Z}$$

where $\overline{x}$ is the class in $S_t^\vee / S_t$ of an element $x \in S^\vee$. By definition, an element

$\overline{x} \in S_t^\vee / S_t$ is said to be isotropic whenever it satisfies

$$q(\overline{x}) = \overline{0} \in \mathbb{Q}/2\mathbb{Z},$$

that is, whenever

$$\langle x, x \rangle_{S_t} \in 2\mathbb{Z}.$$

Let

$$\overline{w} = \overline{\left(\frac{\beta_1}{2t}, \frac{-\beta_2}{2}, \frac{-\beta_3}{2}\right)} \in S_t^\vee / S_t$$

be a non-trivial isotropic element of $S_t^\vee / S_t$. By *non-trivial*, it should be understood that w is not equal to the zero element of

$$(\mathbb{Z}/2t\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z}).$$

That is, we thus assume that

$$\neg\,(2t \mid \beta_1 \text{ and } 2 \mid \beta_2 \text{ and } 2 \mid \beta_3) \tag{2.12}$$

holds. Since $\overline{w}$ is assumed to be an isotropic element of $S_t^\vee / S_t$, the quantity

$$\begin{aligned} q(w) &= \begin{pmatrix} \beta_1/2t & +\beta_2/2 & +\beta/2 \end{pmatrix} \begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} \beta_1/2t \\ \beta_2/2 \\ \beta_3/2 \end{pmatrix} + 2\mathbb{Z} \\ &= (\frac{\beta_1^2}{2t} - \frac{\beta_2^2}{2} - \frac{\beta_3^2}{2}) + 2\mathbb{Z}. \end{aligned}$$

is the zero element of $\mathbb{Q}/2\mathbb{Z}$, that is, there exists an integer $k \in \mathbb{Z}$ such that

$$\frac{\beta_1^2}{2t} - \frac{\beta_2^2}{2} - \frac{\beta_3^2}{2} = 2k \in 2\mathbb{Z}. \tag{2.13}$$

Multiplying both sides of this equality by 2 leads us to

$$\frac{\beta_1^2}{t} = 4k + \beta_2^2 + \beta_3^2. \tag{2.14}$$

Since the right-hand side of this equality is an integer, it is clear that we must have $t \mid \beta_1^2$. Since t is assumed to be strictly greater than two and equal to the product distinct primes, the fact that

$$t \mid \beta_1^2$$

enables us to deduce that

$$t \mid \beta_1$$

after a simple application of Euclid's lemma. The non-triviality condition displayed in, expression (2.12) is a negation of conjunction, and can thus be expressed as a disjunction of negations, i.e.,

$$\neg\,(2t \mid \beta_1\,) \;\text{ or }\; \neg\,(2 \mid \beta_2) \;\text{ or }\; \neg\,(2 \mid \beta_3)$$

from which arise the following seven cases:

(a) $2t \nmid \beta_1,\ 2 \nmid \beta_2,\ 2 \nmid \beta_3$

(b) $2t \nmid \beta_1,\ 2 \nmid \beta_2,\ 2 \mid \beta_3$

(c) $2t \nmid \beta_1,\ 2 \mid \beta_2,\ 2 \nmid \beta_3$

(d) $2t \mid \beta_1,\ 2 \nmid \beta_2,\ 2 \nmid \beta_3$

(e) $2t \mid \beta_1,\ 2 \mid \beta_2,\ 2 \nmid \beta_3$

(f) $2t \nmid \beta_1,\ 2 \mid \beta_2,\ 2 \mid \beta_3$

and

(g) $2t \mid \beta_1,\ 2 \nmid \beta_2,\ 2 \mid \beta_3.$

We assume that at least one of the three β_i is non-zero in each case, so that all the conditions above make sense.

We proceed as follows for the remainder of this section: From each one of the

above-mentioned case, we will exhibit a contradiction and will then be able to assert that an isotropic element

$$\overline{w} = \overline{\left(\frac{\beta_1}{2t}, \frac{-\beta_2}{2}, \frac{-\beta_3}{2}\right)} \in S_t^\vee / S_t,$$

is necessarily trivial whenever t is a product of distinct primes such that

$$t \equiv 3 \pmod 4 .$$

► **Case (a)** - Assume that the conditions

$$2t \nmid \beta_1, 2 \nmid \beta_2, 2 \nmid \beta_3$$

hold. That is, the integers β_2 and β_3 are odd and can respectively be expressed as

$$\beta_2 = 2k_2 + 1 \qquad \text{and} \qquad \beta_3 = 2k_3 + 1.$$

Squaring the expressions for β_2 and β_3, we obtain

$$\beta_2^2 = 4k_2^2 + 4k_2 + 1 \qquad \text{and} \qquad \beta_3^2 = 4k_3^2 + 4k_3 + 1.$$

Feeding these expressions of β_1^2 and β_2^2 into equality (2.13) yields

$$\begin{aligned} \frac{\beta_1^2}{2t} &= 2k + (2k_2^2 + 2k_2 + \frac{1}{2}) + (2k_3^2 + 2k_3 + \frac{1}{2}) \\ &= 2(k + k_2^2 + k_3^2 + k_2 + k_3) + 1. \end{aligned}$$

Multiplying both sides of this equality by $2t$ enables us to obtain that β_1^2 is even. Since $\beta_1 \in \mathbb{Z}$, we immediately obtain that β_1 is even . That is, there exists an integer $n \in \mathbb{Z}$ such that

$$2n = \beta_1$$

We have shown earlier that $t \mid \beta_1$, that is, there exists $m \in \mathbb{Z}$ such that

$$tm = \beta_1$$

Hence, we have

$$2n = tm.$$

Since t is assumed to be a product of distinct primes and such that $t > 2$, there exists $p \in \mathbb{Z}$ such that

$$m = 2p$$

thus

$$2n = 2pm = \beta_1$$

We thus obtained that $2t \mid \beta_1$, which contradicts our initial assumption on β_1.

► **Case (b), Case (c) -** Assume that

$$2t \nmid \beta_1,\ 2 \nmid \beta_2,\ 2 \mid \beta_3 \qquad \text{or that} \qquad 2t \nmid \beta_1, 2 \mid \beta_2, 2 \nmid \beta_3.$$

The fact that $q(\overline{w}) \in 2\mathbb{Z}$ is equivalent to the congruence

$$\beta_1^2/2t - \beta_2^2/2 - \beta_3^2/2 = 0 \pmod 2$$

which, multiplying both sides by $2t$,turns into

$$\beta_1^2 - t\beta_2^2 - t\beta_3^2 \equiv 0 \pmod{4t}. \tag{2.15}$$

Keeping in mind that

$$\mathbb{Z}/(nm)\mathbb{Z} \simeq \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$$

holds if and only if $\gcd(n, m) = 1$, and this formula extends to the case where

more than two primes are involved, that is,

$$\mathbb{Z}/(n_1 n_2 \dots n_r)\mathbb{Z} \simeq (\mathbb{Z}/n_1\mathbb{Z}) \times (\mathbb{Z}/n_2\mathbb{Z}) \times \cdots \times (\mathbb{Z}/n_r\mathbb{Z})\,.$$

Since t is assumed to be equal to a product of distinct primes, we have

$$t = t_1 t_2 \dots t_m$$

for distinct primes t_i with $1 \leq i \leq m$, and hence can write $\mathbb{Z}/4t\mathbb{Z}$ as

$$\mathbb{Z}/\left(4t_1 t_2 \dots t_m\right)\mathbb{Z} \simeq (\mathbb{Z}/4\mathbb{Z}) \times (\mathbb{Z}/t_1\mathbb{Z}) \times \cdots \times (\mathbb{Z}/t_m\mathbb{Z})$$

thus making a $\mathbb{Z}/4\mathbb{Z}$ modular factor apparent. The latter enables us to express (2.15) modulo 4:

$$\beta_1^2 - t\beta_2^2 - t\beta_3^2 \equiv 0 \pmod 4. \tag{2.16}$$

We see that the two following possibilities arise from this congruence:

- Either $t \equiv 3 \pmod 4$
- or $t \equiv 1 \pmod 4$.

First possibility: When $t \equiv 3 \pmod 4$, we have

$$-t \equiv 1 \pmod 4$$

and expression (2.16) turns into

$$\beta_1^2 + \beta_2^2 + \beta_3^2 \equiv 0 \pmod 4. \tag{2.17}$$

All possible modular solutions $(\beta_1, \beta_2, \beta_3)$ of this equation are listed below:

$$\beta_1 \equiv 0, \beta_2 \equiv 0, \beta_3 \equiv 0 \pmod 4,$$

$$\beta_1 \equiv 0, \beta_2 \equiv 0, \beta_3 \equiv 2 \pmod 4,$$

$$\beta_1 \equiv 0, \beta_2 \equiv 2, \beta_3 \equiv 0 \pmod 4),$$

$$\beta_1 \equiv 0, \beta_2 \equiv 2, \beta_3 \equiv 2 \pmod 4),$$

$$\beta_1 \equiv 2, \beta_2 \equiv 0, \beta_3 \equiv 0 \pmod 4),$$

$$\beta_1 \equiv 2, \beta_2 \equiv 0, \beta_3 \equiv 2 \pmod 4),$$

$$\beta_1 \equiv 2, \beta_2 \equiv 2, \beta_3 \equiv 0 \pmod 4),$$

$$\beta_1 \equiv 2, \beta_2 \equiv 2, \beta_3 \equiv 2 \pmod 4).$$

We see that none of above-mentioned solutions $(\beta_1, \beta_2, \beta_3)$ of 2.17 satisfy the conditions

$$2t \nmid \beta_1, 2 \nmid \beta_2, 2 \mid \beta_3$$

of case **(b)**.

Similarly, there is no solution satisfying the conditions

$$2t \nmid \beta_1, 2 \mid \beta_2, 2 \nmid \beta_3$$

of case **(c)**.

Thus, a non-trivial isotropic element satisfying the conditions of cases **(b)** or **(c)** cannot exist when

$$t \equiv 3 \pmod 4).$$

Second possibility: When $t \equiv 1 \pmod 4)$ we have

$$-t \equiv 3 \pmod 4)$$

and expression (2.16) becomes

$$\beta_1^2 + 3\beta_2^2 + 3\beta_3^2 \equiv 0 \pmod 4). \tag{2.18}$$

All possible modular solutions $(\beta_1, \beta_2, \beta_3)$ of this equation are listed below:

$$\beta_1 \equiv 0, \beta_2 \equiv 0, \beta_3 \equiv 0 \pmod 4$$

$$\beta_1 \equiv 0, \beta_2 \equiv 0, \beta_3 \equiv 2 \pmod 4),$$

$$\beta_1 \equiv 0, \beta_2 \equiv 2, \beta_3 \equiv 0 \pmod 4),$$

$$\beta_1 \equiv 0, \beta_2 \equiv 2, \beta_3 \equiv 2 \pmod 4),$$

$$\beta_1 \equiv 1, \beta_2 \equiv 0, \beta_3 \equiv 1 \pmod 4),$$

$$\beta_1 \equiv 1, \beta_2 \equiv 0, \beta_3 \equiv 3 \pmod 4),$$

$$\beta_1 \equiv 1, \beta_2 \equiv 1, \beta_3 \equiv 0 \pmod 4),$$

$$\beta_1 \equiv 1, \beta_2 \equiv 1, \beta_3 \equiv 2 \pmod 4),$$

$$\beta_1 \equiv 1, \beta_2 \equiv 2, \beta_3 \equiv 1 \pmod 4),$$

$$\beta_1 \equiv 1, \beta_2 \equiv 2, \beta_3 \equiv 3 \pmod 4).$$

We see that the solutions

$$\beta_1 \equiv 1, \beta_2 \equiv 1, \beta_3 \equiv 0 \pmod 4 \qquad \text{and} \qquad \beta_1 \equiv 1, \beta_2 \equiv 1, \beta_3 \equiv 2 \pmod 4$$

satisfy the conditions

$$2t \nmid \beta_1, 2 \nmid \beta_2, 2 \mid \beta_3$$

of case **(b)**.

The solutions

$$\beta_1 \equiv 1, \beta_2 \equiv 2, \beta_3 \equiv 1 \pmod 4 \qquad \text{and} \qquad \beta_1 \equiv 1, \beta_2 \equiv 2, \beta_3 \equiv 3 \pmod 4$$

satisfy the conditions

$$2t \nmid \beta_1, 2 \mid \beta_2, 2 \nmid \beta_3$$

of case **(c)**. The existence of isotropic elements is therefore a possibility whenever $t \equiv 1 \pmod 4$ the conditions of cases **(b)** and **(c)** are satisfied. See the following examples.

Example. Assume $t = 13$. Then it is clear that $t \equiv 1 \,(\mathrm{mod}\, 4)$. The integers $\beta_1 = 60437$, $\beta_2 = 90517$ and $\beta_3 = 26316$ satisfy the conditions of case **(b)**, are such that $\beta_1 \equiv 1, \beta_2 \equiv 1, \beta_3 \equiv 0 \,(\mathrm{mod}\, 4)$ and hence satisfy the modular equation (2.18). They thus define an isotropic element of the lattice $S_{13}^{\vee}/\, S_{13}$.

Example. When $t = 17$, we have $t \equiv 1 \,(\mathrm{mod}\, 4)$. The integers $\beta_1 = 44625$, $\beta_2 = 72230$, $\beta_3 = 39285$ satisfy the conditions of case (c), are such that $\beta_1 \equiv 1, \beta_2 \equiv 2, \beta_3 \equiv 1 \,(\mathrm{mod}\, 4)$ and hence satisfy the modular equation (2.18). They therefore define an isotropic element of the lattice $S_{17}^{\vee}/\, S_{17}$.

► **Case (d)** - Assume that the conditions

$$2t \mid \beta_1, 2 \nmid \beta_2, 2 \nmid \beta_3$$

hold. That is, there exist integers $n, k_1, k_2 \in \mathbb{Z}$ such that

$$\beta_1 = 2tn, \quad \beta_2 = 2k_2 + 1 \qquad \text{and} \qquad \beta_3 = 2k_3 + 1.$$

Squaring both sides of each of theses inequalities yields

$$\beta_1^2 = t^2 \cdot 2^2 \cdot n^2, \qquad \beta_2^2 = 4k_2^2 + 4k_2 + 1 \qquad \text{and} \qquad \beta_3^2 = 4k_3^2 + 4k_3 + 1.$$

The equality (2.14) thus becomes

$$2tn^2 = 2k + 2k_2^2 + 2k_2 + 2k_3^2 + 2k_3 + 1. \tag{2.19}$$

Since the left-hand side of this equality is even, while its right-hand side is odd, we see that the assumptions

$$2t \mid \beta_1, 2 \nmid \beta_2, 2 \nmid \beta_3$$

lead us to a contradiction. Thus, an isotropic element defined by $\beta_1, \beta_2, \beta_3$ cannot be non-trivial if the above conditions are satisfied.

► **Case (e)** - Assume that the conditions

$$2t \mid \beta_1, 2 \mid \beta_2, 2 \nmid \beta_3$$

hold. Then there exist integers k_1, k_2 such that

$$\beta_1 = 2tk_1 \qquad \text{and} \qquad \beta_2 = 2k_2.$$

The expression (2.13) can therefore be expressed as

$$\beta_3^2 = 4tk_1^2 - 4k_2^2 - 4k,$$

and we deduce that β_3^2 an even integer. Since the square of an odd integer is necessarily odd, it is clear β_3 cannot be odd.

Hence $2 \mid \beta_3$, contradicting our initial assumption on β_3.

► **Case (f)** - Assume that the conditions

$$2t \nmid \beta_1, 2 \mid \beta_2, 2 \mid \beta_3$$

hold. There exist integers k_1, k_2 such that

$$\beta_2 = 2k_1 \qquad \text{and} \qquad \beta_3 = 2k_2.$$

The equality (2.13) can then be turned into

$$\beta_1^2 = 4tk_1^2 + 4k_2^2 + 4tk,$$

thus making apparent the fact that β_1^2 an even integer, that is, $2 \mid \beta_1^2$, which in

turns implies that

$$2 \mid \beta_1.$$

Keeping in mind that $t \mid \beta_1$ always hold, we hence see that

$$2t \mid \beta_1,$$

thus contradicting our initial assumption $2t \nmid \beta_1$.

► **Case (g)** - Assume that the conditions

$$2t \mid \beta_1, 2 \nmid \beta_2, 2 \mid \beta_3$$

hold. There exist integers k_1, k_2 such that

$$\beta_1 = 2tk_1 \qquad \text{and} \qquad \beta_3 = 2k_3.$$

The equality (2.13) can then be turned into

$$\beta_2^2 = 4tk_1^2 - 4k_3^2 - 4k,$$

thus making apparent the fact that β_2^2 an even integer. As indicated earlier, the square of an odd integer is necessarily odd. Thus β_2 cannot be odd. We therefore deduce that $2 \mid \beta_2$, contradicting our initial assumption on β_2. Note that β_2 and β_3 have a symmetric role in expression 2.13 and in cases (e) and (f), hence the proofs for these two cases follow the same pattern. We hence established that whenever t is assumed to be equal to a product of distinct primes greater such that

$$t \equiv 3 \pmod 4,$$

then assuming the existence of a **non-trivial** isotropic element $\overline{w} \in S_t^\vee / S_t$ leads to a contradiction. Hence, the discriminant group $S_t^\vee / S_t$ of $S_t = \mathrm{NS}(X_t)$ has no non-trivial isotropic elements whenever the integer parameter t satisfies the above-mentioned conditions.

2.5 About dimension of linear systems

Let k be an algebraically closed field. Denote by $\mathbb{P}^n$ the n-dimensional projective space over k. It is well-known that for any integer $d > 0$, there is a bijection between the linear system of hypersurfaces of degree d in $\mathbb{P}^n$ and the projectivization of the set

$$H^0(\mathbb{P}^n, \mathcal{O}(d))$$

of global sections of $\mathcal{O}(d)$. That is, there is a bijection between the linear system of hypersurfaces of degree d in $\mathbb{P}^n$ and set of degree d homogenenous polynomials. The linear system Γ_d^n of hypersurfaces of degree d in $\mathbb{P}^n$ has therefore projective dimension equal to

$$\begin{aligned} \dim\ \Gamma_d^n &= \dim\ H^0(\mathbb{P}^n, \mathcal{O}(d)) - 1 \\ &= \binom{d+n}{n} - 1. \end{aligned}$$

Points $s_0, s_1, \ldots, s_{r-1}$ in $\mathbb{P}^n$ are said to be in *general position* whenever the following conditions are satisfied:

- If $r < n + 1$, then the vectors defined by the homogenous coordinates of these r points are linearly independent.
- for $r = n + 1$, any n points are linearly independent.

Assume that $s_0, s_1, \ldots, s_{r-1}$ are r points in general position in $\mathbb{P}^n$.

Remark. From now on until the end of this thesis, all curves are considered general, and in general position. When defining a curve, for instance, a curve C in $\mathbb{P}^3$, one should start by fixing a certain number of points in general position in $\mathbb{P}^3$ and then require that C passes through them so that a hypersurface containing the points must contain the curve. All the curves involved should thus be defined by imposing that they pass through a sufficiently low number of generic points. Additionally, intersections are always supposed transverse.

The following statements hold:

- The linear system $\Gamma_d^n(s_0, \ldots, s_{r-1})$ of hypersurfaces of degree d in $\mathbb{P}^n$ containing the point $s_0, \ldots, s_{r-1}$ has dimension

$$\dim\ \Gamma_d^n(s_0, \ldots, s_{r-1}) = \binom{n+d}{n} - 1 - r.$$

- The linear system $\Gamma_d^n(C)$ of hypersurfaces of degree d in $\mathbb{P}^n$ containing a general curve C of degree m has dimension given by the formula

$$\dim\ \Gamma_d^n(C) = \binom{n+d}{n} - 1 - (m \cdot d + 1).$$

- The linear system $\Gamma_d^n(C_0, C_1)$ of hypersurfaces of degree d in $\mathbb{P}^n$ containing two general curves C_0 and C_1 of degree m intersecting transversely has dimension

$$\dim\ \Gamma_d^n(C_0, C_1) = \binom{n+d}{n} - 1 - (2(m \cdot d + 1) - C_0 \cdot C_1).$$

- The linear system $\Gamma_d^n(C_0, \ldots, C_{r-1})$ of hypersurfaces of degree d in $\mathbb{P}^n$ containing general curves $C_0, \ldots, C_{r-1}$ of degree m intersecting transversely has dimension

$$\dim\ \Gamma_d^n(C_0, \ldots, C_{r-1}) = \binom{n+d}{n} - 1 - (r(m \cdot d + 1) - (\sum_{i<j} C_i \cdot C_j)).$$

Example 40. Let C_1, C_2 be two disjoint conics in $\mathbb{P}^3$. The linear system Γ_1 of quartics containing C_1 and C_2 is 16 dimensional. Indeed, we have

$$\dim\ \Gamma_4^3(C_1, C_2) = \binom{4+3}{3} - 1 - 2 \cdot (2 \cdot 4 + 1) = 35 - 1 - 18 = 16$$

More details and examples can be found by clicking here.

2.6 Computer-based study of projective models and unirationality of moduli spaces

We now explain how we made use of the material introduced in the previous sections to study projective models of $K3$ surfaces. In order to deal with our initial objective, which consisted in studying projective models and the unirationality of the moduli spaces of $K3$ surfaces with Néron-Severi group isomorphic to the integral lattice with Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix},$$

we produced solutions that turned out to have a much wider scope of application. The following result, that can be traced back to Morisson's 1988 Cortona summer lectures with elements from Saint-Donat [17] and stated below in its form due to Debarre in his lectures [3], is of great importance for our study:

Theorem 41. ***(SDM - Saint-Donat / Morrison)*** *Let X be a $K3$ surface and let $D \in \mathrm{NS}(X)$ be an ample class.*

(a) If $D^2 = 2$ and there does not exist a class $F \in \mathrm{NS}(X)$ such that $F^2 = 0$ and $F \cdot D = 1$ then $\varphi_D : X \longrightarrow \mathbb{P}^2$ is a double cover.

(b) If $D^2 = 4$ and there does not exist a class $F \in \mathrm{NS}(X)$ on X such that $F^2 = 0$ and $F \cdot D \in \{1, 2\}$ then $\varphi_D : X \longrightarrow \mathbb{P}^3$ embeds X as a quartic surface in $\mathbb{P}^3$.

(c) If $D^2 = 6$ and there does not a divisor F on X such that $F^2 = 0$ and $F \cdot D \in \{1, 2\}$ then $\varphi_D : X \longrightarrow \mathbb{P}^4$ embeds X as a degree 6 surface in $\mathbb{P}^4$.

(d) If $D^2 = 8$ and there does not exist a class F on X such that $F^2 = 0$ and $F \cdot D \in \{1, 2, 3\}$ then $\varphi_D : X \longrightarrow \mathbb{P}^5$ either embeds X as a generically transverse intersection of three quadrics in $\mathbb{P}^5$ with only rational double points, or φ_D realizes X as double cover of a Veronese surface.

This theorem is fascinating because it provides precise and explicit numerical criteria and conditions which must be fulfilled in order for an ample class on a $K3$ to be associated with a projective model of this surface . Had we had the opportunity to travel back in time to 1988, we would probably have to face to the fact that using such a theorem with some degree of automation would have been quite difficult. Indeed, decades ago, the state of technology did not allow researchers to mobilize hardware endowed with the processing power that we enjoy today. This theorem is often used in its classical and equivalent formulation, and it is even still the case today. This formulation, which involves the notions of base-point freeness and non-hyperellipticity, was probably favored by researchers at the time. The two formulations, classic and modern, of the theorem, are nevertheless logically equivalent. Indeed, various results which can be traced back to Saint-Donat state that given an ample class $D \in \mathrm{NS}(X)$, the non-existence of classes F such that $F^2 = 0$ and $F \cdot D = 1$ is equivalent to the base-point freeness of D. Likewise, for classes such that $D^2 \geq 4$, establishing the non-hyperellipticity of D ensures that there does not exist a class $F \in \mathrm{NS}(X)$ such that $F \cdot D = 2$. We can therefore assume without taking a considerable risk that, in the past, in order to make use of the vintage **SDM** theorem, people had to:

- Handcraft base-point freeness, ampleness and non-hyperellipticity criteria specific to each $K3$ surface under study.
- Find a class $D \in \mathrm{NS}(X)$ satisfying these criteria.

Doing so was without any doubt not an easy task, and all these constraints reduced the possibilities of study to a handful of cases. Almost four decades later, the situation is radically different. Nothing stands in the way of full automation:

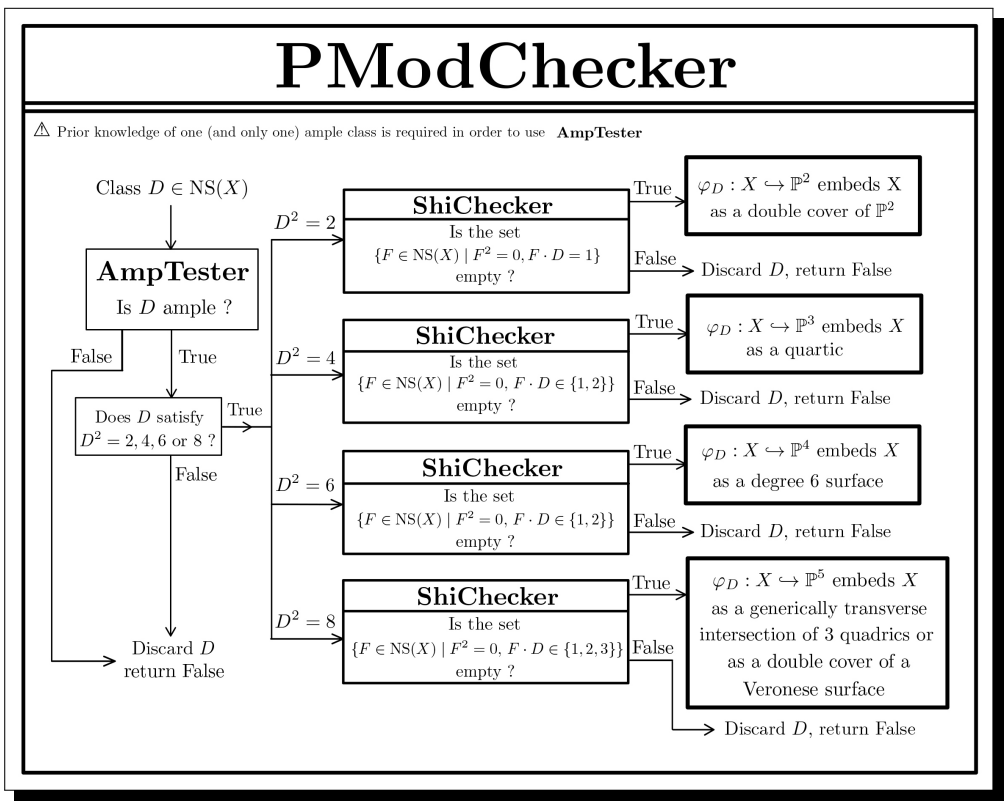
- The program **CGS** is capable of producing an abundance of data on classes $D \in \mathrm{NS}(X)$ of any desired self-intersection D^2.
- We can determine whether a class $D \in \mathrm{NS}(X)$ is ample using the program **AmpTester**.

Thus, the procedure **CGS** from section 2.1 enables us to obtain data on classes $D \in \mathrm{NS}(X)$ of divisors of self-intersection $D^2 = 2, 4, 6$ or 8, while the procedure **AmpTester** from section 2.2 enables us to identify ample classes among the data on classes produced by the procedure **CGS**. The only requirement to be fulfilled to execute this strategy consists in finding an initial ample class. We show in the section 2.3 of this thesis how this can be done. We, moreover, have material to deal with conditions of existence or non-existence of classes of divisors F on X such that $F^2 = 0$, $F \cdot D \in \{1, 2, 3\}$. Indeed, given an ample class $D \in \mathrm{NS}(X)$ and integers $n_1, n_2 > 0$, the procedure **ShiChecker** detailed in section 2.2.1 is capable of computing sets of the form

$$\{F \in \mathrm{NS}(X) \mid \langle F, D\rangle = n_1,\ \langle F, F\rangle = n_2\}\,.$$

We, therefore, have in our hands all the necessary ingredients to give life to the **SDM** theorem: We can now determine whether any class $D \in \mathrm{NS}(X)$ can be associated with a projective model of X in virtue of this theorem. The resulting tool is **PModChecker**, for Projective Models Checker. We introduce and explain how to use this tool on our website. Assuming given a Gram matrix of $\mathrm{NS}(X)$ with respect to a fixed basis and an ample class $a_0 \in \mathrm{NS}(X)$ as ambient parameters, **PModChecker** takes as input a class $D \in \mathrm{NS}(X)$ and determines whether it fits within the framework of one of the cases of the **SDM** theorem. When this is the case, it returns the precise information on the nature of the projective model which can be obtained from the knowledge of the class D.

The structure of the procedure **PModChecker** can be illustrated as follows :



We return to our initial objective: Exhibiting values of t and conditions under which a quartic in $\mathbb{P}^3$ with Néron-Severi group isomorphic to $\mathrm{NS}(X_t)$ can be built from scratch, that is, establishing the unirationality of the moduli space of the surfaces X_t, for these values of t. Reaching this goal requires the addition of a geometrical flavor to our approach. To do so, we use Roulleau's technique from his articles [16] and [15] as a starting point. In order to study projective models of a $K3$ surface while putting emphasis on a genuine geometric aspect, Roulleau enforces a technique which consists in:

- Establishing criteria of non-hyperellipticity and base-point freeness for classes in $\mathrm{NS}(X)$, to then apply the vintage **SDM** theorem.
- Using the data produced by his program **SmoothRationalCurves** to handcraft a configuration of smooth rational curves associated with an ample, base-point free and non-hyperelliptic class.

As discussed earlier, our program **PModChecker** enables us to disregard all considerations involving the notions of non-hyperellipticity and base-point freeness by using numerical criteria instead. We thus focus on the second point. A prototypical example of the configurations found in Roulleau's atlas of $K3$ surfaces [16] is of the following type:

$$\begin{cases} C_1 + C_2 = n_1 D \\ C_3 + C_4 = n_2 D \end{cases} \tag{2.20}$$

where the class D is ample, n_1, n_2 are positive integers and C_1, C_2, C_3, C_4 are distinct classes in $\mathrm{NS}(X)$ of smooth rational curves on X. Such a configuration can be formalized by introducing the notion of *system*:

Definition 42. Let D be an ample class. We use the term **system** to refer to a finite collection $\{\mathcal{L}_j\}$ of linear combinations of classes of smooth rational curves each satisfying $\mathcal{L}_i = n_i D$ for some positive integer n_i with the additional properties that:

- All linear combinations are made of the same number of (-2)-curves.
- All curves involved in a linear combination are distinct.
- No class of smooth rational curve (-2)-curve can be involved in more than one linear combination.

The definition of a system has a wide scope and encompasses many types of configurations, such as a configuration made of a single linear combination involving three classes of (-2)-curves, e.g.,

$$C_1 + C_2 + C_3 = nD$$

or configurations with three linear combinations and four classes of (-2)-curves per linear combination, e.g.,

$$\begin{cases} C_1 + C_2 + C_3 + C_4 & = & n_1 D \\ C_5 + C_6 + C_7 + C_8 & = & n_2 D \\ C_9 + C_{10} + C_{11} + C_{12} & = & n_3 D \end{cases}$$

and many other possible forms. There are so many possibilities that we have introduce a precisely defined framework to purse our study.

We follow Roulleau's steps by focusing on systems involving two linear combinations, each made of two classes of (-2)-curves per linear combination, that is:

$$\begin{cases} C_1 + C_2 = n_1 D \\ C_3 + C_4 = n_2 D \end{cases} \tag{2.21}$$

In order to obtain such systems on a $K3$ surface, we use our program **SysFinder**,

detailed and available for download on K3surfaces.com. From the input data of a Gram matrix of the Néron-Severi group $\mathrm{NS}(X)$ of a $K3$ surface, of an ample class $a_0 \in \mathrm{NS}(X)$, and of an integer $c > 0$, our program **SysFinder** takes advantage of the procedure **CGS** to produce data on classes of smooth rational curves and on classes of divisors having squares $2, 4, 6$ or 8. The program **SysFinder** then calls for **AmpTester** to identify ample classes and finally processes all this data to exhibit systems of the form (2.21) . Assume that a system

$$\begin{cases} C_1 + C_2 = n_1 D \\ C_3 + C_4 = n_2 D \end{cases} \tag{2.22}$$

with $D^2 = 4$ has thus been obtained. By definition 42 of a system, $D \in \mathrm{NS}(X)$ is assumed to be ample. Assume moreover than an application of **PModChecker** with D as input data returned that $\varphi_D : X \hookrightarrow \mathbb{P}^3$ realizes X as a quartic in $\mathbb{P}^3$. We now explain how the data of a system can lead to the explicit construction of such a quartic. First, note that each linear combination which is part of a system can be viewed as a sub-system of the system under study:

- The sub-system

  $$C_1 + C_2 = n_1 D \qquad \text{(sub-system \textbf{I})}$$

  may be realized in $\mathbb{P}^3$ as the intersection of a quartic surface with a hypersurface of degree n_1. When this is the case, such an intersection can be expressed as the union of curves A_1 and A_2 such that

  $$\deg(A_1) = C_1 \cdot D \qquad \text{and} \qquad \deg(A_2) = C_2 \cdot D.$$

- Similarly, the sub-system

  $$C_3 + C_4 = n_2 D \qquad \text{(sub-system \textbf{II})}$$

  may be realized in $\mathbb{P}^3$ as the intersection of a quartic surface with a hy-

persurface of degree n_2. When this is the case, this intersection then decomposes as the union of curves A_3 and A_4 such that

$$\deg(A_3) = C_3 \cdot D \qquad \text{and} \qquad \deg(A_4) = C_4 \cdot D.$$

It would be convenient to construct both sub-systems **I** and **II** in such a way that the respective intersections they define are both contained on the same quartic surface $\mathcal{Q}$ in $\mathbb{P}^3$ and in such a way that all the A_i are smooth rational curves, i.e.,

$$A_i \simeq \mathbb{P}^1 \quad \text{for} \quad i \in \{1, 2, 3, 4\}\,.$$

To this end, we proceed as follows: Let $A_1 \simeq \mathbb{P}^1$ and $A_3 \simeq \mathbb{P}^1$ be rational normal curves in $\mathbb{P}^3$ having respectively degree

$$\deg(A_1) = C_1 \cdot D \qquad \text{and} \qquad \deg(A_3) = C_3 \cdot D$$

and satisfying

$$A_1 \cdot A_3 = C_1 \cdot C_3.$$

We check whether there exists a quartic in $\mathbb{P}^3$ containing A_1 and A_3 by computing the projective dimension of the linear system of quartic surfaces in $\mathbb{P}^3$ containing the curves A_1 and A_3 and checking whether this dimension is superior or equal to zero. We thus introduce the **Condition LS1**:

$$\binom{4+3}{3} - 1 - (4\deg(A_1) + 1) - (4\deg(A_3) + 1) + C_1 \cdot C_3 \geq 0$$

Assume that **LS1** is satisfied and pick a quartic $\mathcal{Q}$ in the above-mentioned linear system. By intersecting $\mathcal{Q}$ with a degree n_1 hypersurface H_1 containing the curve A_1, we produce a residual rational normal curve $A_2 \simeq \mathbb{P}^1$ such that

$$A_1 + A_2 = n_1 H_1,$$

thus mimicking sub-system **I** within of a quartic $\mathbb{P}^3$. However, we first have

to determine whether the linear system of surfaces of degree n_1 containing the curve A_1 has a projective dimension superior or equal to zero. This is **Condition LS2:**

$$\binom{n_1+3}{3} - 1 - (n_1 \deg(A_1) + 1) \geq 0$$

Assume that **Condition LS2** holds. We still have to find a curve $A_4 \simeq \mathbb{P}^1$ in $\mathbb{P}^3$ which will play the role of the curve associated with the class C_4. This can be done by intersecting $\mathcal{Q}$ with a degree n_2 section containing C_3, thus producing a residual rational normal curve $A_4 \simeq \mathbb{P}^1$ such that

$$\deg(A_4) = C_4 \cdot D.$$

As before, such an operation can only be performed when the linear system of surfaces of degree n_2 containing the curve A_3 has a projective dimension superior or equal to zero. This is **Condition LS3:**

$$\dim\, \Gamma(\mathbb{P}^3, n_2 \mid A_3) = \binom{n_2+3}{3} - 1 - (n_2 \deg(A_3) + 1) \geq 0$$

When conditions **LS1**, **LS2** and **LS3** hold, it can be established that the Néron-Severi group $\mathrm{NS}(\mathcal{Q})$ of the quartic $\mathcal{Q}$ surface thus constructed in $\mathbb{P}^3$ contains a copy of the Néron-Severi group $\mathrm{NS}(X)$ of the surface under study, i.e.,

$$\mathrm{NS}(X) \subseteq \mathrm{NS}(\mathcal{Q}).$$

Before proceeding further, note that conditions **LS1**, **LS2**, and **LS3** only depend on parameters that can be obtained from the data of the system under study. Our program **SystemFinder** is capable of identifying systems satisfying these three conditions and discard the others. If we show that the discriminant group of $\mathrm{NS}(X)$ does not contain non-trivial isotropic elements, then the result mentioned at the beginning of section 2.4 enables us to deduce that $\mathrm{NS}(X)$ cannot have a proper overlattice, i.e.,

$$\mathrm{NS}(\mathcal{Q}) \simeq \mathrm{NS}(X)$$

hence establishing the unirationality of the moduli space of $K3$ surfaces with Néron-Severi group $\mathrm{NS}(X)$ due to the explicit construction of the quartic performed in projective space. Indeed, constructing a surface such as X amounts to constructing rational normal curves A_1 and A_3 in $\mathbb{P}^3$ with prescribed intersection value $C_1 \cdot C_3$ and then taking a quartic in the linear system of quartic surfaces containing them if the latter is non-empty. Such a construction can be realized as a result of conditions **LS1**, **LS2** and **LS3** being assumed to hold. This construction is moreover done with rational parameters. We enforced this strategy in order to study the family of surfaces X_t with Néron-Severi group isomorphic to the integral lattice with Gram matrix

$$\begin{pmatrix} 2t & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

with respect to a fixed basis, where we restricted to cases for which the positive integer parameter t satisfies $t \equiv 3 \pmod 4$ and can be expressed as a product of distinct primes.

- **SysFinder** is used to generate systems of the form (2.22) each associated with a class D with $D^2 = 4$ and satisfying conditions **LS1**, **LS2** and **LS3**.
- Such classes are tested against the **SDM** theorem with **PModChecker** so that only systems associated with classes D such that $\varphi_D : X_t \hookrightarrow \mathbb{P}^3$ realizes X_t as a quartic are considered, and all others discarded.

Recall that **PModChecker** integrates **AmpTester**. Thus, determining whether any given class is ample or not ample can be done without hassle. Assume that the positive integer t_0 is chosen in such a way as to satisfy $t_0 \equiv 3 \pmod 4$ and as being expressible as a product of distinct primes.

Assume that a system satisfying all the conditions mentioned above has been found. We show on K3surfaces.com that a quartic $\mathcal{Q}$ in $\mathbb{P}^3$ such that

$$\mathrm{NS}(X_{t_0}) \subseteq \mathrm{NS}(\mathcal{Q})$$

can then be constructed. From the assumption that t_0 satisfies $t_0 \equiv 3 \pmod 4$ and is a product of distinct primes, proposition 38 enables us to assert that the discriminant group of $\mathrm{NS}(X_{t_0})$ has no isotropic elements, so that $\mathrm{NS}(X_{t_0})$ has no overlattice. In this case, we obtain

$$\mathrm{NS}(X_{t_0}) \simeq \mathrm{NS}(\mathcal{Q})$$

and are then able to assert the unirationality of the moduli space of $K3$ surfaces with Néron-Severi group isomorphic to $\mathrm{NS}(X_{t_0})$.

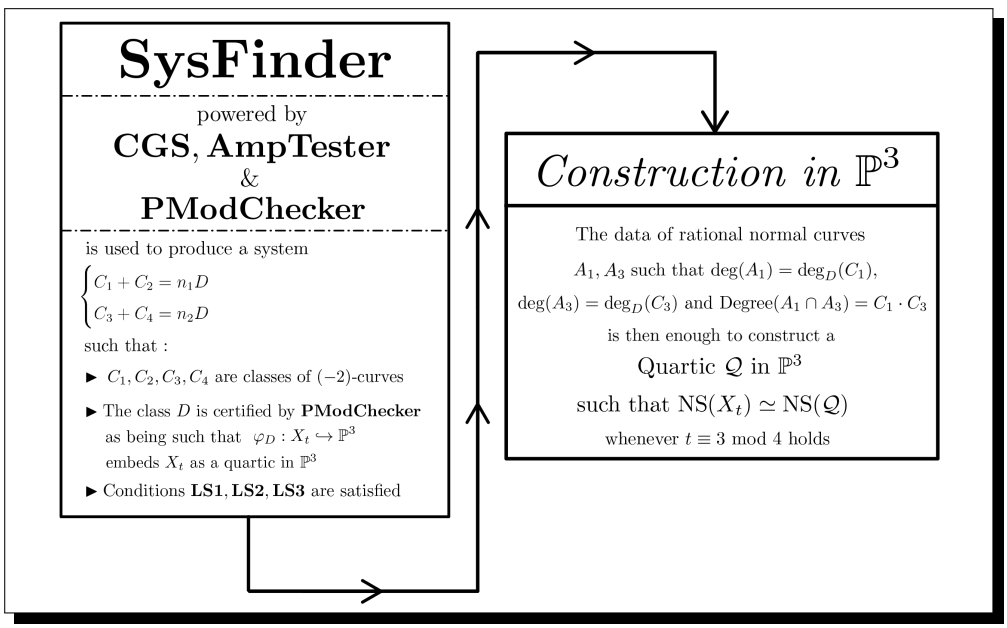
Note that the approach we used regarding unirationality is fully compliant with the strategy devised by Professor Xavier Roulleau to do so. We thus have the duty to emphasize the fact that we merely applied his methods, and that the innovation lies in the fact that we enforced them using a computer-based algorithmic approach and determined conditions and concrete tools to exhibit explicit constructions leading to unirationality in the framework of the family of $K3$ surfaces X_t, whose automorphism groups and orbits of smooth rational curves had to be studied in order to achieve this doctoral project. In practice, checking whether these conditions indeed hold amounts to finding a suitable system satisfying **LS1**, **LS2**, **LS3** with **SysFinder** (which involves **CGS**, **PModChecker** and **AmpTester**) with the additional requirements that the integer parameter t must satisfy $t \equiv 3 \pmod 4$ and can be expressed as a product of distinct primes. When this is the case, we have

$$\mathrm{NS}(X_t) \simeq \mathrm{NS}(Y)$$

where Y is the quartic constructed in $\mathbb{P}^3$.

Note that finding a suitable system is the purpose of our program **SysFinder** from the **proj_mod** suite.

The overall procedure can be summarized as indicated in the following figure:



More details about the practical and computer-based side of this procedure can be found as additional online content. We illustrate the methods and techniques presented in this section by using the case of the $K3$ surface X_7 as an example. This content can be accessed by clicking here. One last time, we have to mention that dealing with the computer-based aspect of this thesis cannot be done in a conventional manuscript. We kindly ask our readers to keep in mind that K3surfaces.com has been created to make up for the limitations of this PDF file.

A detailed table containing all the references used in this thesis can be found by clicking here.

All the figures used in this thesis
can be found in high resolution by clicking here.

A table summarizing all the procedures related to Borcherds' method can be found by clicking here.

References

[1] Richard Borcherds. Automorphism groups of Lorentzian lattices. *Journal of Algebra*, 111(1):133–153, 1987.

[2] Richard Borcherds. Coxeter groups, Lorentzian lattices, and K3 surfaces. *International Mathematics Research Notices, Volume 1998, Issue 19, Pages 1011-1031*, 1998.

[3] Olivier Debarre. Surfaces K3 - Graduate Course - Spring 2019.

[4] Igor Dolgachev and S. Kondō. A supersingular K3 surface in characteristic 2 and the Leech lattice. *International Mathematics Research Notices*, 2003, 01 2002.

[5] Daniel Huybrechts. *Lectures on K3 Surfaces*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2016.

[6] Jonghae Keum and S. Kondō. The automorphism groups of Kummer surfaces associated with the product of two elliptic curves. *Transactions of the American Mathematical Society*, 353:1469–1487, 01 2001.

[7] Shigeyuki Kondō. Algebraic $K3$ surfaces with finite automorphism groups. *Nagoya Mathematical Journal*, 116:1 – 15, 1989.

[8] Shigeyuki Kondō. The automorphism group of a generic Jacobian Kummer surface. *Journal of Algebraic Geometry*, 7:589–609, 1998.

[9] Pierre Lairez and Emre Can Sertöz. A numerical transcendental method in algebraic geometry : Computation of Picard groups and related invariants. *SIAM Journal on Applied Algebra and Geometry*, 3(4):559–584, 2019.

[10] Ujikawa Masashi. The automorphism group of the singular K3 surface of discriminant 7. 2013.

[11] Curtis McMullen. K3 surfaces, entropy and glue. *Journal für die Reine und Angewandte Mathematik*, 658, 09 2009.

[12] Giacomo Mezzedimi. *Elliptic K3 surfaces and their moduli dynamics, geometry and arithmetic.* PhD thesis, Hannover : Gottfried Wilhelm Leibniz Universität,, 2021.

[13] David R. Morrison. The geometry of K3 surfaces. *Lectures delivered at the Scuola Matematica Interuniversitaria, Cortona, Italy, July 31 - August 27*, 1988.

[14] I. R. Piatetskii-Shapiro, I. Shafarevich. A Torelli theorem for algebraic surfaces of type K3. *Math. USSR Izv., 35 (1971) 530-572*, 1971.

[15] Xavier Roulleau. On the geometry of K3 surfaces with finite automorphism group and no elliptic fibrations. 2019.

[16] Xavier Roulleau. An atlas of K3 surfaces with finite automorphism group. April 2020.

[17] Bernard Saint-Donat. Projective models of K-3 surfaces. *American Journal of Mathematics*, 96(4):602–639, 1974.

[18] Ichiro Shimada. Projective models of the supersingular K3 surface with Artin invariant 1 in characteristic 5. *Journal of Algebra*, 403:273–299, 2014.

[19] Ichiro Shimada. An algorithm to compute automorphism groups of K3 surfaces and an application to singular K3 surfaces. *International Mathematics Research Notices*, 2015.

[20] Hans Sterk. Finiteness results for algebraic K3 surfaces. *Mathematische Zeitschrift*, 189:507–514, 1985.

[21] E. B. Vinberg. Some arithmetic discrete groups in Lobachevskii spaces. *"Discrete sub groups of Lie groups and applications to Moduli", Tata-Oxford (1975), 323–348.*, 1975.