Hermitian Modular Forms for Fields of Low Discriminant

Diploma Thesis
in Mathematics

by
Albert Zeyer

submitted to the
Faculty of Mathematics, Computer Science and Natural Science of
RWTH Aachen University

October 2012
revised version from June 10, 2013

Supervisor: Prof. Dr. Aloys Krieg
Second examiner: Dr. Martin Raum

written at the
Lehrstuhl A für Mathematik
Prof. Dr. A. Krieg

Contents

Chapter 1

Introduction

We develop an algorithm to compute Fourier expansions of Hermitian modular forms of degree 2 over $\mathrm{Sp}_2(\mathcal{O})$ for $\mathcal{O} \subseteq \mathbb{Q}(\sqrt{-\Delta})$, $\Delta \in \{3, 4, 8\}$.

In [PY07], spaces of Siegel modular cusp forms are calculated. It uses a linear reduction of Siegel modular forms to Elliptic modular forms and gains information from there. This is very similar to what we are doing with Hermitian modular forms.

A similar algorithm is also [Rau12, Algorithm 4.3] for Jacobi forms.

We are doing the same for Hermitian modular forms.

Along with the theoretical work, the algorithm has also been implemented. The implementation has been done with the Sage ([S$^+$13]) framework. It is implemented in C++ ([Str83]), Cython ([BBS$^+$13]) and Python ([vR13]). The code can be found on GitHub ([Zey13a]) and another backup might be on [Zey13b].

Chapter 2

Preliminaries

$\mathbb{N}$ denotes the set $\{1, 2, 3, \ldots\}$, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ and $\mathbb{Z}$ are all **integers**. $\mathbb{Q}$ are all the **rational numbers**, $\mathbb{R}$ are the **real numbers** and $\mathbb{C}$ are the **complex numbers**. $\mathbb{R}^+ := \{x \in \mathbb{R} \mid x > 0\}$, $\mathbb{R}^\times$ and $\mathbb{C}^\times$ denotes all non-zero numbers.

Let $\mathrm{Mat}_n(R)$ be the set of all $n \times n$ **matrices** over some commutative ring R. Likewise, $\mathrm{Mat}_n^T(R)$ are the **symmetric** $n \times n$ matrices. X^T is the **transposed** matrix of $X \in \mathrm{Mat}_n(R)$. $\overline{Z}$ is the **conjugated** matrix of $Z \in \mathrm{Mat}_n(\mathbb{C})$. For $R \subseteq \mathbb{C}$, $\overline{R} \subseteq R$, the set of **Hermitian matrices** in R is defined as

$$\mathrm{Her}_n(R) = \left\{ Z \in \mathrm{Mat}_n(R) \,\middle|\, \overline{Z}^T = Z \right\}.$$

Thus, for $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{Her}_n(R)$, we have $a = \overline{a}$ and $d = \overline{d}$, i.e. $a, d \in R \cap \mathbb{R}$. We also have $c = \overline{b}$ and thus we introduce the shorter notation (**Gauß notation**) $[a, b, c] := \left(\begin{smallmatrix} a & b \\ \overline{b} & c \end{smallmatrix}\right) \in \mathrm{Her}_n(\mathbb{C})$ for $a, c \in \mathbb{R}$, $b \in \mathbb{C}$.

A matrix $Y \in \mathrm{Mat}_n(\mathbb{C})$ is greater 0 if and only if $\forall x \in \mathbb{C}^n - \{0\} : Y[x] := \overline{x}^T Y x \in \mathbb{R}^+$. Such matrices are called the **positive definitive matrices**, defined by

$$\mathcal{P}_n(R) = \{X \in \mathrm{Mat}_n(R) \mid X > 0\}$$

for $R \subseteq \mathbb{C}$. Note that $\mathcal{P}_n(R) \subseteq \mathrm{Her}_n(R)$, i.e. all positive definite matrices are Hermitian. For a matrix over $\mathbb{R}$, it means that it is also symmetric.

For $A, X \in \mathrm{Mat}_n(\mathbb{C})$, we define $A[X] := \overline{X}^T A X$. The **denominator** of a matrix $Z \in \mathrm{Mat}_n(\mathbb{Q})$ is the smallest number $x \in \mathbb{N}$ such that $xZ \in \mathrm{Mat}_n(\mathbb{Z})$. We also write $\mathrm{denom}(Z) = x$. $1_n \in \mathrm{Mat}_n(\mathbb{Z})$ denotes the **identity matrix**.

The **general linear group** is defined by

$$\mathrm{GL}_n(R) = \{X \in \mathrm{Mat}_n(R) \mid \det(X) \text{ is a unit in } R\}$$

and the **special linear group** by

$$\mathrm{SL}_n(R) = \{X \in \mathrm{Mat}_n(R) \mid \det(X) = 1\}.$$

The **orthogonal group** is defined by

$$\mathrm{O}_n(R) = \left\{ X \in \mathrm{GL}_n(R) \,\middle|\, X^T 1_n X = 1_n \right\} \subseteq \mathrm{GL}_n(R).$$

The **symplectic group** is defined by

$$\mathrm{Sp}_n(R) = \left\{ X \in \mathrm{GL}_{2n}(R) \,\middle|\, \overline{X}^T J_n X = J_n \right\} \subseteq \mathrm{GL}_{2n}(R) \subseteq \mathrm{Mat}_{2n}(R)$$

where $J_n := \left(\begin{smallmatrix} 0 & -1_n \\ 1_n & 0 \end{smallmatrix}\right) \in \mathrm{SL}_{2n}(R)$ (as in [Der01]). (Note that some authors (e.g. [PY07]) define J_n negatively.) $\mathrm{Sp}_n(R)$ is also called the **unitary group**. Note that [Der01] uses $\mathrm{U}_n(R) = \mathrm{Sp}_n(R)$. Also note that $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathbb{Z}) \Leftrightarrow ad - bc = 1 \Leftrightarrow M \in \mathrm{SL}_2(\mathbb{Z})$. Thus, $\mathrm{Sp}_1(\mathbb{Z}) = \mathrm{SL}_2(\mathbb{Z})$.

In addition, for a ring $R \subseteq \mathbb{C}$, define

$$\begin{aligned} \mathrm{Rot}(U) &:= \begin{pmatrix} \overline{U}^T & \\ & U^{-1} \end{pmatrix} \in \mathrm{Sp}_2(R), & U \in \mathrm{GL}_2(R) \\ \mathrm{Trans}(H) &:= \begin{pmatrix} 1_2 & H \\ & 1_2 \end{pmatrix} \in \mathrm{Sp}_2(R), & H \in \mathrm{Her}_2(R) \end{aligned}$$

and note that we have $J_2 = \left(\begin{smallmatrix} & -1_2 \\ 1_2 & \end{smallmatrix}\right) \in \mathrm{Sp}_2(R)$. Those tree types of matrices form a generator set for the group $\mathrm{Sp}_2(R)$.

For $Z \in \mathrm{Mat}_n(\mathbb{C})$, we call

$$\Re(Z) := \frac{1}{2}\left(Z + \overline{Z}^T\right) \in \mathrm{Mat}_n(\mathbb{C})$$

the **real** part and

$$\Im(Z) := \frac{1}{2i}\left(Z - \overline{Z}^T\right) \in \mathrm{Mat}_n(\mathbb{C})$$

the **imaginary** part of Z and we have $Z = \Re(Z) + i\Im(Z)$. Note that we usually have $\Re(Z), \Im(Z) \notin \mathrm{Mat}_n(\mathbb{R})$ but we have $\Re(Z), \Im(Z) \in \mathrm{Her}_n(\mathbb{C})$.

We say that some function $f\colon \mathcal{A} \to \mathcal{B}$ with $\mathcal{A} \subseteq \mathrm{Mat}_n(R)$, $\mathcal{B} \subseteq R$ is k**-invariant** under some $\mathcal{X} \subseteq \mathrm{Mat}_n(R)$ where $\mathcal{A}[\mathcal{X}] := \{A[X] \in \mathcal{A} \mid X \in \mathcal{X}\} \subseteq \mathcal{A}$ if and only if

$$\det(U)^k f(T[U]) = f(T) \text{ for all } T \in \mathcal{A}, U \in \mathcal{X}.$$

We write

$$\left(\mathcal{B}^{\mathcal{A}}\right)^{\mathcal{X}} := \left\{ f \in \mathcal{B}^{\mathcal{A}} \,\middle|\, f \text{ is } k\text{-invariant under } \mathcal{X} \right\}.$$

2.1 Elliptic modular forms

Elliptic modular forms are holomorphic functions over the set

$$\mathcal{H}_1 := \{z \in \mathbb{C} \mid \Im(z) > 0\} \subseteq \mathbb{C}$$

which is called the **Poincaré upper half plane**.

Let f be a holomorphic function $\mathcal{H}_1 \to \mathbb{C}$. **Modular forms** are functions which are invariant with regard to a specific **translation**. In this case, the translation is given by some $M \in \mathrm{Sp}_1(\mathbb{Z}) = \mathrm{SL}_2(\mathbb{Z})$ and a **weight** $k \in \mathbb{Z}$.

Let $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathbb{Z})$ and $\tau \in \mathcal{H}_1$. We write

$$M\tau := \frac{a\tau + b}{c\tau + d}.$$

Note that we have $\Im(M\tau) = \frac{\Im(\tau)}{(c\Re(\tau)+d)^2+(c\Im(\tau))^2} > 0$ and thus $M\tau \in \mathcal{H}_1$. We define the **translated function** $f|M \colon \mathcal{H}_1 \to \mathbb{C}$ as

$$(f|_k M)(\tau) := (c\tau + d)^{-k} \cdot f(M\tau).$$

If the weight k is fixed in the context, we also write $f|M := f|_k M$.

Let Γ be a subgroup of $\mathrm{Sp}_1(\mathbb{Z})$. We also call Γ the **translation group**.

An **Elliptic modular form** with weight $k \in \mathbb{Z}$ over Γ is a holomorphic function

$$f \colon \mathcal{H}_1 \to \mathbb{C}$$

with

$$\begin{aligned} &(1)\ \ f|M = f \quad \forall\, M \in \Gamma, \\ &(2)\ \ f(\tau) = O(1) \quad \text{for } \tau \to i\infty. \end{aligned}$$

Thus, (1) yields the equation

$$f\left(\frac{a\tau + b}{b\tau + c}\right) = (c\tau + d)^k \cdot f(\tau) \quad \forall\ \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \Gamma, \tau \in \mathcal{H}_1.$$

$\mathcal{M}_k(\Gamma)$ denotes the vector space of such Elliptic modular forms.

In this work, we use a specific subgroup of $\mathrm{Sp}_1(\mathbb{Z})$. We define

$$\Gamma_0(l) := \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{Sp}_1(\mathbb{Z}) \,\middle|\, c \equiv 0 \pmod{l} \right\} \subseteq \mathrm{Sp}_1(\mathbb{Z}) \subseteq \mathrm{Mat}_2(\mathbb{Z})$$

as a subgroup of $\mathrm{Sp}_1(\mathbb{Z})$.

An **Elliptic modular cusp form** is an Elliptic modular form $f \colon \mathcal{H}_1 \to \mathbb{C}$ with

$$\lim_{t \to \infty} f(it) = 0.$$

We can represent the cusps with $\Gamma \backslash \mathbb{Q}$.

2.2 Siegel modular forms

Siegel modular forms are a generalization of Elliptic modular forms for higher dimensions. Let

$$\mathcal{H}_n := \left\{ Z \in \mathrm{Mat}_n^T(\mathbb{C}) \,\middle|\, \Im(Z) > 0 \right\}$$

be the **Siegel upper half space**. We call $\mathrm{Sp}_n(\mathbb{Z})$ the **Siegel modular group**. Siegel modular forms are holomorphic functions $\mathcal{H}_n \to \mathbb{C}$ for a given **degree** $n \in \mathbb{N}$.

The **translation group** Γ is a subgroup of $\mathrm{Sp}_n(\mathbb{Z})$. For $M = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) \in \mathrm{Sp}_n(\mathbb{Z})$ and $Z \in \mathcal{H}_n$, we write

$$M \cdot Z := (AZ + B) \cdot (CZ + D)^{-1}.$$

Again, we can confirm that $M \cdot Z \in \mathcal{H}_n$. Generalizing the Elliptic translation, the Siegel **translated function** $f|M \colon \mathcal{H}_n \to \mathbb{C}$ is defined as

$$(f|M)(Z) := \det(CZ + D)^{-k} \cdot f(M \cdot Z)$$

A **Siegel modular form** of degree $n \in \mathbb{N}$ with weight $k \in \mathbb{Z}$ over Γ is a holomorphic function

$$f \colon \mathcal{H}_n \to \mathbb{C}$$

with

(1) $f|M = f \quad \forall\, M \in \Gamma$,
(2) for $n = 1$: $f(Z) = O(1) \quad$ for $Z \to i\infty$

$\mathcal{M}_k^{\mathcal{H}_n}(\Gamma)$ denotes the vector space of such Siegel modular forms.

Note that Elliptic modular forms are Siegel modular forms of degree $n = 1$. Thus we have $\mathcal{M}_k(\Gamma) = \mathcal{M}_k^{\mathcal{H}_1}(\Gamma)$.

Siegel modular forms aren't directly used in this work. However, the idea of this work is inspired by [PY07] and they are using them.

2.3 Hermitian modular forms

Let

$$\mathbb{H}_n := \{ Z \in \mathrm{Mat}_n(\mathbb{C}) \mid \Im(Z) > 0 \}$$

be the **Hermitian upper half space**. Note that these matrices are not symmetric as the Siegel upper half space $\mathcal{H}_n$ but we have $\mathcal{H}_n \subseteq \mathbb{H}_n$ and $\mathcal{H}_1 = \mathbb{H}_1 \subseteq \mathbb{C}$.

Hermitian modular forms are holomorphic functions $\mathbb{H}_n \to \mathbb{C}$. They are a generalization of Siegel modular forms where the **translation group** Γ is not a subgroup of $\mathrm{Sp}_n(\mathbb{Z})$ but a subgroup of $\mathrm{Sp}_n(\mathcal{O})$ for some $\mathcal{O} \subseteq \mathbb{C}$.

More specificially, let $\Delta \in \mathbb{N}$ so that we have the imaginary quadratic number field $\mathbb{K} := \mathbb{Q}(\sqrt{-\Delta})$ where $-\Delta$ is the fundamental discriminant. Then, let $\mathcal{O} \subseteq \mathbb{Q}(\sqrt{-\Delta})$ be the maximum order. We call $\mathrm{Sp}_n(\mathcal{O})$ the **Hermitian modular group**. Let Γ be a subgroup of $\mathrm{Sp}_n(\mathcal{O})$.

Again, with $M = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) \in \mathrm{Sp}_n(\mathcal{O})$, $Z \in \mathbb{H}_n$, $M \cdot Z := (AZ + B) \cdot (CZ + D)^{-1} \in \mathbb{H}_n$ as for Siegel modular forms and the **weight** $k \in \mathbb{Z}$, we define the **translated function** $f|M \colon \mathbb{H}_n \to \mathbb{C}$ as

$$(f|M)(Z) := \det(CZ + D)^{-k} \cdot f(M \cdot Z).$$

A **Hermitian modular form** of **degree** $n \in \mathbb{N}$ with **weight** $k \in \mathbb{Z}$ over Γ is a holomorphic function

$$f \colon \mathbb{H}_n \to \mathbb{C}$$

with

(1) $f|M = f \quad \forall\, M \in \Gamma, Z \in \mathbb{H}_n$,

(2) for $n = 1$: f is holomorphic in all cusps.

$\mathcal{M}_k^{\mathbb{H}_n}(\Gamma)$ denotes the vector space of such Hermitian modular forms.

As it can be done for Siegel modular forms, we generalize this further by introducing a **Multiplicative character** $\nu \colon \Gamma \to \mathbb{C}^\times$. Thus, for $M_1, M_2 \in \Gamma$, we have $\nu(M_1) \cdot \nu(M_2) = \nu(M_1 \cdot M_2)$.

A **Hermitian modular form** over Γ and ν is a holomorphic function

$$f \colon \mathbb{H}_n \to \mathbb{C}$$

with

(1) $f|M = \nu(M) \cdot f \quad \forall\, M \in \Gamma, Z \in \mathbb{H}_n$,

(2) for $n = 1$: f is holomorphic in all cusps.

$\mathcal{M}_k^{\mathbb{H}_n}(\Gamma, \nu)$ denotes the vector space of such Hermitian modular forms.

For $f \in \mathcal{M}_k^{\mathbb{H}_n}(\Gamma, \nu)$, define the **Siegel Φ-operator** as

$$(f|\Phi)(Z') := \lim_{t\to\infty} f\begin{pmatrix} Z' & 0 \\ 0 & it \end{pmatrix}, \quad Z' \in \mathbb{H}_{n-1}.$$

Then (see [Der01]), $f|\Phi\colon \mathbb{H}_{n-1} \to \mathbb{C}$ is a well-defined Hermitian modular form of degree $n-1$.

A Hermitian modular form $f \in \mathcal{M}_k^{\mathbb{H}_n}(\Gamma, \nu)$ is a **Hermitian modular cusp form**, if and only if for all $R \in \mathrm{Sp}_n(\mathbb{K})$, it holds

$$(f|R)|\Phi \equiv 0.$$

In this work, we will always use Hermitian modular forms of degree $n = 2$.

2.3.1 Properties

Because $-\Delta$ is fundamental, we have two possible cases:

1. $\Delta \equiv 3 \pmod 4$ and Δ is square-free, or
2. $\Delta \equiv 0 \pmod 4$, $\Delta/4 \equiv 1, 2 \pmod 4$ and $\Delta/4$ is square-free.

And for the **maximal order** $\mathcal{O}$, we have (compare [Der01])

$$\begin{aligned} \mathcal{O} &= \mathbb{Z} + \mathbb{Z}\frac{-\Delta + i\sqrt{\Delta}}{2}, \\ \mathcal{O}^{\#} &= \mathbb{Z}\frac{i}{\sqrt{\Delta}} + \mathbb{Z}\frac{1 + i\sqrt{\Delta}}{2}. \end{aligned}$$

From now on, we will always work with Hermitian modular forms of degree $n = 2$. We also use $\Gamma = \mathrm{Sp}_2(\mathcal{O})$ for simplicity.

Chapter 3

Theory

In this section, we will develop the theoretical foundation for the tools to calculate the space of Fourier expansions of some precision of Hermitian modular forms $\mathcal{FE}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma))$.

We know that there is a basis of Fourier expansions such that all Fourier coefficients are over $\mathbb{Q}$.

We start with the space of all possible Fourier expansions, i.e. with the space $\mathcal{M}_0 := \mathbb{Q}^{\mathcal{I}}$ for some index set $\mathcal{I}$. The tools in this section are all some specific conditions which lead to some vectorspace $\tilde{\mathcal{M}} \subset \mathbb{Q}^{\mathcal{I}}$ which are always superspaces of $\mathcal{FE}_{\mathcal{I}}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma)) \subset \mathbb{Q}^{\mathcal{I}}$. Thus, when intersecting such space, we iteratively get new subspaces

$$\mathcal{M}_{i+1} := \mathcal{M}_i \cap \tilde{\mathcal{M}}.$$

With other methods, we know the dimension of $\mathcal{FE}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma))$. Thus we can easily determine whether $\mathcal{M}_i = \mathcal{FE}_{\mathcal{I}}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma))$, i.e. whether we are finished and can terminate the algorithm.

It is not proven that this series of spaces eventually gets to $\mathcal{FE}_{\mathcal{I}}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma))$ but from other research, this seems likely.

3.1 Restriction to Elliptic modular forms

We develop the first method to calculate a vectorspace $\tilde{\mathcal{M}} \subset \mathcal{FE}(\mathcal{M}_n^{\mathbb{H}_2}(\Gamma))$. This methods works by restricting Hermitian modular forms to Elliptic modular forms. A similar method to restrict Siegel modular forms to Elliptic modular forms has been presented in [PY07].

Methods to calculate the vectorspace of Elliptic modular forms are well known and thus we gain information by this restriction.

We start by describing the restriction.

Lemma 3.1 (Restriction). *Let $f : \mathbb{H}_2 \to \mathbb{C}$ be a Hermitian modular form of weight k with $\nu \equiv 1$. Let $S \in \mathcal{P}_2(\mathcal{O})$. Then, $\tau \mapsto f(S\tau) : \mathbb{H}_1 \subseteq \mathbb{C} \to \mathbb{C}$ is an Elliptic modular form of weight $2k$ to $\Gamma_0(l)$, where l is the denominator of S^{-1}.*

We write

$$f[S]\colon \mathbb{H}_1 \to \mathbb{C}, \quad \tau \mapsto f(S\tau).$$

Proof. Define $\Gamma^H := \mathrm{Sp}_2(\mathcal{O})$ as the translation group for f. Let $\tau \in \mathbb{H}_1$. With $S = [s,t,u] \in \mathcal{P}_2(\mathbb{C})$ we have

$$\begin{aligned}\Im(S\tau) &= \frac{1}{2i}\left(S\tau - \overline{S}^T\overline{\tau}\right)\\ &= \frac{1}{2i}S(\tau-\overline{\tau})\\ &= \frac{1}{2i}S\cdot 2i\Im(\tau)\\ &= S\Im(\tau) > 0,\end{aligned}$$

thus $S\tau \in \mathbb{H}_2$. Thus, $\tau \mapsto f(S\tau)$ is a function $\mathbb{H}_1 \to \mathbb{C}$.

Let $l := \det(S)$. That is the denominator of S^{-1}. Let $\left(\begin{smallmatrix} a & b \\ c & d\end{smallmatrix}\right) \in \Gamma_0(l) \subseteq \mathrm{SL}_2(\mathbb{Z})$. We have

$$\begin{aligned}&S\frac{a\tau+b}{c\tau+d}\\ =\;& (a(S\tau)+bS)\cdot((cS^{-1})(S\tau)+d)^{-1}\\ =\;& \begin{pmatrix} a1_2 & bS \\ cS^{-1} & d1_2\end{pmatrix}\cdot S\tau.\end{aligned}$$

Define

$$M := \begin{pmatrix} a1_2 & bS \\ cS^{-1} & d1_2\end{pmatrix} \in \mathrm{Mat}_4(\mathbb{C}).$$

With $l|c$, we also have $cS^{-1} = \frac{c}{l}[u,-t,s] \in \mathrm{Mat}_2(\mathcal{O})$, thus we have $M \in \mathrm{Mat}_4(\mathcal{O})$. Recall that we have $S = \overline{S}^T$ and $ad - bc = 1$. Verify that we have $M \in \mathrm{Sp}_2(\mathcal{O}) = \Gamma^H$:

$$\begin{aligned}&\overline{M}^T J_2 M\\ =\;& \overline{\begin{pmatrix} a1_2 & bS \\ cS^{-1} & d1_2\end{pmatrix}}^T J_2 \begin{pmatrix} a1_2 & bS \\ cS^{-1} & d1_2\end{pmatrix}\\ =\;& \begin{pmatrix} (-acS^{-1} + ac\overline{S^{-1}}^T) & (-ad1_2 + cb\overline{S^{-1}}^T S) \\ (-bc\overline{S}^T S^{-1} + ad1_2) & (-bd\overline{S}^T + bdS)\end{pmatrix}\\ =\;& J_2.\end{aligned}$$

Thus, because f is a Hermitian modular form, we have

$$\begin{aligned}
& f[S]\left(\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)\tau\right) \\
= \; & f\left(S\frac{a\tau+b}{c\tau+d}\right) \\
= \; & f\left(M \cdot S\tau\right) \\
= \; & \nu(M) \cdot \det(cS^{-1}S\tau + d1_2)^k \cdot f(S\tau) \\
= \; & (c\tau+d)^{2k} \cdot f[S](\tau).
\end{aligned}$$

This is the same as

$$(f[S])|_{2k}\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) = f[S].$$

It follows that $f[S]$ is an Elliptic modular form of weight $2k$ to $\Gamma_0(l)$. $\square$

Remark 3.2. Let us analyze the case $\nu \not\equiv 1$. According to [Der01], only for $\Delta \equiv 0 \pmod 4$, there is a single non-trivial Abel character ν. This ν has the following properties (see [Der01]):

$$\begin{aligned}
\nu(J_2) &= 1, \\
\nu(\mathrm{Trans}(H)) &= (-1)^{h_1+h_4+|h_2|^2}, && H = [h_1, h_2, h_4] \in \mathrm{Her}_2(\mathcal{O}) \\
\nu(\mathrm{Rot}(U)) &= (-1)^{|1+u_1+u_4|^2|1+u_2+u_3|^2+|u_1u_4|^2} && U = \left(\begin{smallmatrix} u_1 & u_2 \\ u_3 & u_4 \end{smallmatrix}\right) \in \mathrm{GL}_2(\mathcal{O})
\end{aligned}$$

Consider the proof of the previous lemma. To calculate $\nu(M)$ with the given equations, we need to represent M in the generating system J_2, $\mathrm{Trans}(H)$ and $\mathrm{Rot}(U)$.

We must consider two different cases. Recall that we have $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$, i.e. $ad - bc = 1$, $S = [s, t, u] \in \mathcal{P}_2(\mathcal{O})$ and

$$M = \begin{pmatrix} a1_2 & bS \\ cS^{-1} & d1_2 \end{pmatrix} \in \mathrm{Sp}_2(\mathcal{O}).$$

Case 1: $c = 0$. Then we have $ad = 1$. Define $T := \frac{b}{d}S$. Then we have

$$\begin{aligned}
& \operatorname{Trans}\left(\frac{b}{d}S\right)\operatorname{Rot}\left(\frac{1}{d}1_2\right) \\
= & \begin{pmatrix} 1_2 & \frac{b}{d}S \\ & 1_2 \end{pmatrix}\begin{pmatrix} \frac{1}{d}1_2 & \\ & d1_2 \end{pmatrix} \\
= & \begin{pmatrix} \frac{1}{d}1_2 & bS \\ & d1_2 \end{pmatrix} \\
= & M.
\end{aligned}$$

And we have

$$\begin{aligned}
\nu\left(\operatorname{Trans}\left(\frac{b}{d}S\right)\right) &= (-1)^{\frac{b}{d}s+\frac{b}{d}u+\left|\frac{b}{d}t\right|^2}, \\
\nu\left(\operatorname{Rot}\left(\frac{1}{d}1_2\right)\right) &= (-1)^{\left|1+\frac{2}{d}\right|^2+\left|\frac{1}{d^2}\right|^2} = 1.
\end{aligned}$$

Case 2: $c \neq 0$. Then we have

$$\begin{aligned}
& \operatorname{Trans}\left(\frac{a}{c}S\right)\operatorname{Rot}\left(-\frac{1}{c}S\right)(-J_2)\operatorname{Trans}\left(-\frac{d}{c}S\right)^{-1} \\
= & \begin{pmatrix} 1_2 & \frac{a}{c}S \\ & 1_2 \end{pmatrix}\begin{pmatrix} -\frac{1}{c}\overline{S}^T & \\ & -cS^{-1} \end{pmatrix}(-J_2)\begin{pmatrix} 1_2 & -\frac{d}{c}S \\ & 1_2 \end{pmatrix}^{-1} \\
= & \begin{pmatrix} -\frac{1}{c}\overline{S}^T & -a1_2 \\ & -cS^{-1} \end{pmatrix}\begin{pmatrix} & 1_2 \\ -1_2 & \end{pmatrix}\begin{pmatrix} 1_2 & \frac{d}{c}S \\ & 1_2 \end{pmatrix} \\
= & \begin{pmatrix} -\frac{1}{c}\overline{S}^T & a1_2 \\ & -cS^{-1} \end{pmatrix}\begin{pmatrix} & 1_2 \\ -1_2 & -\frac{d}{c}S \end{pmatrix} \\
= & \begin{pmatrix} a1_2 & -\frac{1}{c}\overline{S}^T + \frac{ad}{c}S \\ cS^{-1} & d1_2 \end{pmatrix} \\
= & M.
\end{aligned}$$

And we have

$$\begin{aligned}
\nu\left(\mathrm{Trans}\left(\frac{a}{c}S\right)\right) &= (-1)^{\frac{a}{c}s+\frac{a}{c}u+\left|\frac{a}{c}t\right|^2},\\
\nu\left(\mathrm{Rot}\left(-\frac{1}{c}S\right)\right) &= (-1)^{\left|1-\frac{1}{c}s-\frac{1}{c}u\right|^2\left|1-\frac{2}{c}\Re(t)\right|^2+\left|\frac{su}{c^2}\right|^2},\\
\nu\left(-J_2\right) &= -1,\\
\nu\left(\mathrm{Trans}\left(-\frac{d}{c}S\right)\right)^{-1} &= (-1)^{-\frac{d}{c}s-\frac{d}{c}u+\left|\frac{d}{c}t\right|^2}.
\end{aligned}$$

As a conclusion for now, it looks complicated to restrict $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, i.e. the translation group Γ^E for the Elliptic modular forms, to satisfy $\nu(M) = 1$. For example, for the case $c = 0$, one fulfilling condition would be $2|b$.

To avoid such complications, we will use $\nu \equiv 1$ for the rest of our work. □

Preliminaries 3.3. We want to calculate a generating set for the Fourier expansions of Hermitian modular forms.

We define the index set

$$\Lambda := \left\{ 0 \leq \begin{pmatrix} a & b \\ \bar{b} & c \end{pmatrix} \in \mathrm{Mat}_2(\mathcal{O}^{\#}) \,\middle|\, a, c \in \mathbb{Z} \right\}$$

as the index for the Fourier coefficients of the Fourier expansions of our Hermitian modular forms.

For a holomorphic function $f\colon \mathbb{H}_2 \to \mathbb{C}$, we write its Fourier expansion as

$$f(Z) = \sum_{T \in \Lambda} a(T) \cdot e^{2\pi i \cdot \mathrm{tr}(TZ)}$$

with its Fourier coefficients $a\colon \Lambda \to \mathbb{Q}$.

Note that some authors use $e^{\pi i}$ as the coefficient bases and define the index set Λ in such a way that tr$(TS) \in 2\mathbb{Z}$ for $T \in \Lambda$ and $S \in \mathrm{Her}_2(\mathcal{O})$. In that case, T is called "even" and one would only allow even matrices in Λ. We don't do that and we keep the factor 2 in the coefficient base, i.e. we use $e^{2\pi i}$.

Remark 3.4. For any $S \in \mathcal{P}_2(\mathcal{O})$, for the restricted function $f[S]\colon \mathbb{H}_1 \to \mathbb{C}$, this gives us

$$f[S](\tau) = \sum_{T \in \Lambda} a(T) \cdot e^{2\pi i \cdot \mathrm{tr}(TS\tau)} = \sum_{T \in \Lambda} a(T) \cdot e^{2\pi i \cdot \mathrm{tr}(TS)\tau}.$$

We use $a[S]\colon \mathbb{N}_0 \to \mathbb{Q}$ for the Fourier coefficients of $f[S]$, i.e. we have

$$f[S](\tau) = \sum_{n \in \mathbb{N}_0} a[S](n) \cdot e^{2\pi i n \tau}.$$

This gives us

$$a[S](n) = \sum_{T \in \Lambda, \mathrm{tr}(ST) = n} a(T).$$

For the implementation of the algorithm, we need to define a finite precision of the index set of the Fourier coefficients of the Hermitian modular forms. Fix $B := B_{\mathcal{F}} \in \mathbb{N}$ as a limit. Define the precision of the Fourier coefficient index

$$\mathcal{F} := \mathcal{F}_B := \left\{ \begin{pmatrix} a & b \\ \bar{b} & c \end{pmatrix} \in \Lambda \,\middle|\, 0 \le a, c < B_{\mathcal{F}} \right\} \subseteq \Lambda.$$

The main algorithm is going to be described in Algorithm 3.13. It will start with the vectorspace of all possible Fourier expansions for the precision index set $\mathcal{F}$ and reduce that vectorspace.

Lemma 3.5. *Given a Hermitian modular form f and its Fourier expansion coefficients $a\colon \mathcal{F}_B \to \mathbb{Q}$ of the precision index set $\mathcal{F}_B$ and a matrix $S = [s, t, u] \in \mathcal{P}_2(\mathcal{O})$, the precision of the Fourier expansion of the Elliptic modular form $f[S]$ is given by*

$$\mathcal{F}(S) = B \cdot (s + u - 2\,|t|),$$

i.e. we can calculate the Fourier expansion coefficients (as described in remark 3.4)

$$a[S]\colon \{k \in N_0 \mid k < \mathcal{F}(S)\} \to \mathbb{Q}.$$

Proof. For a given $S \in \mathcal{S}$ and limit $B \in \mathbb{N}$ which restricts $\mathcal{F} \subset \Lambda$, $\mathcal{F}(S) \in \mathbb{N}_0$ is the limit such that for any $T \in \Lambda - \mathcal{F}$, $\mathrm{tr}(ST) \ge \mathcal{F}(S)$. Thus, for calculating the Fourier coefficients $T \in \Lambda$ with $\mathrm{tr}(ST) \in \{0, \ldots, \mathcal{F}(S) - 1\}$, it is sufficient to enumerate the $T \in \mathcal{F}$.

Let $S = [s, t, u]$ and $T = [a, b, c]$. Recall that $S \in \mathcal{P}_2(\mathcal{O})$. Then we have

$$\mathrm{tr}(ST) = as + \bar{t}b + t\bar{b} + cu = as + cu + 2\Re(\bar{t}b).$$

Because $T \geq 0$, we have $ac \geq |b|^2$ and thus

$$|b| \leq \sqrt{ac} \leq \max(a, c).$$

Thus,

$$2\Re(\bar{t}b) \geq -2\,|t|\,|b| \geq -2\,|t| \max(a, c).$$

We also have $as + cu \geq \max(a, c)(s + u)$. Assuming $T \in \Lambda - \mathcal{F}$, we have $\max(a, c) \geq B$. For such T, we get

$$\text{tr}(ST) \geq B \cdot (s + u - 2\,|t|).$$

Given $S > 0$, we have $su > |t|^2$. Then we have

$$\begin{aligned} & s + u - 2\,|t| > 0 \\ \Leftrightarrow \quad & su + u^2 - 2\,|t|\,u > 0 \\ \Leftrightarrow \quad & \left(|t|^2 + u^2 - 2\,|t|\,u\right) + (su - |t|^2) > 0 \\ \Leftrightarrow \quad & (|t| - u)^2 + (su - |t|^2) > 0. \end{aligned}$$

Thus, for $B > 0$, we have

$$B \cdot (s + u - 2\,|t|) > 0.$$

All inequalities were sharp estimations[1], thus we get

$$\mathcal{F}(S) = B \cdot (s + u - 2\,|t|).$$ □

Remark 3.6. Let $\mathcal{M}_i$ be a sub vector space of Hermitian modular form Fourier expansions $a\colon \mathcal{F} \to \mathbb{Q}$, i.e. $\mathcal{M}_i \subset \mathcal{FE}_{\mathcal{F}}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma))$. Remark 3.4 and lemma 3.5 gives us the tools to reduce $\mathcal{M}_i$ to a sub vector space $\mathcal{M}_{i+1} \subset \mathcal{M}_i$. Note that we don't neccessarily have strict reductions. In many cases, $\mathcal{M}_{i+1} = \mathcal{M}_i$.

For a given $S \in \mathcal{P}_2(\mathcal{O})$, when calculating the restrictions $a \mapsto a[S]$ for all $a \in \mathcal{M}_i$, we must only get Fourier expansions of Elliptic modular forms. In remark 3.8, we will see how to calculate the restricted Elliptic modular form Fourier expansions $a[S]$. And we can

[1] For example, let $S = [2, -1, 1]$. Then you have $s + u - 2\,|t| = 1$. With $c = B$ and $a = b = 1$, you hit the limit $\text{tr}(ST) = 2 + B - 2 = B = \mathcal{F}(S)$.

independently calculate the space of Elliptic modular form Fourier expansions and thus calculate the new space.

Thus,

$$\mathcal{M}_{i+1} := \left\{a \in \mathcal{M}_i \;\middle|\; a[S] \in \mathcal{FE}_{\mathcal{F}(S)}(\mathcal{M}_k(\Gamma_0(l_S)))\right\} \cup \{a \in \mathcal{M}_i \mid a[S] \equiv 0\}.$$

Remark 3.7. In the algorithm, we want to work with Fourier expansions in $\mathbb{Q}^{\mathcal{F}}$. A canonical basis is the set $\mathcal{F}$. We analyze how practical this is in a Computer implementation.

With $[a, b, c] \in \mathcal{F}$, we have $0 \leq a, c < B$, thus there are only a finite number of possible $(a, c) \in \mathbb{N}_0^2$. Because $0 \leq [a, b, c]$, we get $ac - |b|^2 \geq 0$ and thus b is also always limited. Thus, $\mathcal{F}$ is finite but it might be huge for even small B. For example[2],

$$\text{for } D = -3, B = 10, \quad \text{we have } \#\mathcal{F} = 21892.$$

$$\text{for } D = -3, B = 20, \quad \text{we have } \#\mathcal{F} = 413702.$$

Because we want $a \in \mathbb{Q}^{\mathcal{F}}$ to be a Fourier expansions of Hermitian modular forms, we can assume that a is invariant under $\mathrm{GL}_2(\mathcal{O})$. This means that we have

$$\det(U)^k a(T[U]) = a(T) \quad \forall\, U \in \mathrm{GL}_2(\mathcal{O}),$$

where k is the weight of the Hermitian modular forms. This is the set $\left(\mathbb{Q}^{\mathcal{F}}\right)^{\mathrm{GL}_2(\mathcal{O})}$, i.e. all the Fourier expansions which satisfy this invariation. In our algorithm, we can work with that set instead if we want to calculate Hermitian modular forms.

Let us develop a basis of $\left(\mathbb{Q}^{\mathcal{F}}\right)^{\mathrm{GL}_2(\mathcal{O})}$: For $T_1, T_2 \in \mathcal{F}$, define the equivalence relation

$$T_1 \sim_{\mathrm{GL}_2(\mathcal{O})} T_2 \quad \Leftrightarrow \quad \exists\, U \in \mathrm{GL}_2(\mathcal{O})\colon\; T_1[U] = T_2.$$

Thus, we can identify a basis of $\left(\mathbb{Q}^{\mathcal{F}}\right)^{\mathrm{GL}_2(\mathcal{O})}$ by $\mathcal{F}/{\sim_{\mathrm{GL}_2(\mathcal{O})}}$. We use the same invariation notation as for $\mathbb{Q}^{\mathcal{F}}$ and write

$$\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})} := \mathcal{F}/{\sim_{\mathrm{GL}_2(\mathcal{O})}}.$$

We identify the elements in $\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}$ by reduced matrices[3] in $\mathcal{F}$. Then, we have $(\mathcal{F})^{\mathrm{GL}_2(\mathcal{O})} \subseteq \mathcal{F}$.

[2] This example was calculated with the code at [Zey13a].

[3] The matrices are reduced in some sense of Minkowski. Details can be seen in section 4.5 and in the source code at [Zey13a]. There is an algorithm which, for a given matrix $T \in \mathcal{F}$, calculates a reduced matrix $\tilde{T}$ and a determinant character det such that $\tilde{T}[U] = T$ for some $U \in \mathrm{GL}_2(\mathcal{O})$ with $\det(U) = e^{2\pi i \cdot det/\#(\mathcal{O}^\times)}$.

Restricting the elements in $\mathcal{F}$ by the $\mathrm{GL}_2(\mathcal{O})$-invariation makes the set $(\mathcal{F})^{\mathrm{GL}_2(\mathcal{O})} \subseteq \mathcal{F}$ much smaller and better to handle in Computer calculations. For example,

$$\text{for } D = -3, B = 10, \quad \text{we have } \# \left(\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}\right) = 420,$$
$$\text{for } D = -3, B = 20, \quad \text{we have } \# \left(\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}\right) = 4840.$$

This makes the set $\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}$, to identify a basis of the finite dimension vector space $\left(\mathbb{Q}^{\mathcal{F}}\right)^{\mathrm{GL}_2(\mathcal{O})}$, much more practical to be used in a Computer implementation. □

Remark 3.8. From remark 3.4 and lemma 3.5, we have

$$a[S](i) = \sum_{T \in \mathcal{F}, \mathrm{tr}(ST) = i} a(T)$$

for $i \in \mathbb{N}_0, i < \mathcal{F}(S)$.

Set $N := \# \left(\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}\right)$ and let

$$\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})} = \{T_1, \ldots, T_N\}$$

where T_j are the reduced matrices in $\mathcal{F}$.

We have

$$\det(U)^k a(T_j[U]) = a(T_j)$$

for all $j \leq N, U \in \mathrm{GL}_2(\mathcal{O})$, where k is the weight of the Hermitian modular form. For any $T \in \mathcal{F}$, we can (see also remark 3.7 and section 4.5) uniquely find $j_T \leq N$ and $U_T \in \mathrm{GL}_2(\mathcal{O})$ such that

$$T_{j_T}[U_T] = T.$$

Then we have

$$a(T) = a(T_{j_T}[U]) = \det(U_T)^{-k} a(T_{j_T}).$$

Thus,

$$a[S](i) = \sum_{T \in \mathcal{F}, \mathrm{tr}(ST) = i} \det(U_T)^{-k} a(T_{j_T}).$$

This gives us the formula to calculate the Fourier expansion $a[S] \colon \{i \in \mathbb{N}_0 \mid i < \mathcal{F}(S)\} \to \mathbb{Q}$.

This is a linear map $\mathbb{Q}^{\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}} \to \mathbb{Q}^{\{i\in\mathbb{N}_0 \mid i<\mathcal{F}(S)\}} \cong \mathbb{Q}^{\mathcal{F}(S)}$ and the matrix $M \in \mathrm{Mat}_{\mathcal{F}(S)\times N}(\mathbb{Q})$ of this map is given by the formula above. When we identify

$$a = (a(T_1), \ldots, a(T_N)),$$
$$a[S] = (a[S](0), \ldots, a[S](\mathcal{F}(S)-1)),$$

then the i-th row and the j-th column is given by

$$M_{i,j} = \sum_{T\in\mathcal{F},\mathrm{tr}(ST)=i,j_T=j} \det(U_T)^{-k}$$

and we have

$$M \cdot a = a[S].$$

The implementation of the calculation of M and its details are described in section 4.7. □

3.2 Elliptic modular cusp forms

Restricting the space of Hermitian modular form Fourier expansion space via remark 3.6 is probably not enough.

Another method is to use information from cusp forms. We can restrict them in a similar way as earlier and we get relations between the restrictions and the space of Elliptic modular cusp forms.

Preliminaries 3.9. Let $\hat{c} \in \mathbb{Q} \cup \{\infty\}$ be a representation of a cusp in $\Gamma_0(l)$. For our method, we are only interested in $\hat{c} \neq \infty$. We choose a matrix $M = M_{\hat{c}} = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{SL}_2(\mathbb{Z})$ such that $M\infty = \hat{c}$. We show in lemma 3.10 how to do that. M is also called a cusp matrix.

Let $f\colon \mathbb{H}_2 \to \mathbb{C}$ be a Hermitian modular form and $S \in \mathcal{P}_2(\mathcal{O})$. Now we look at $f[S]|_{2k}M$.

We have

$$\begin{aligned}
& (f[S]|_{2k}M)(\tau) \\
= \; & (c\tau + d)^{-2k} \cdot f[S](M\tau) \\
= \; & \det(c1_2\tau + d1_2)^{-k} \cdot f(S(M\tau)) \\
= \; & \det(c1_2\tau + d1_2)^{-k} \cdot f((aS\tau + bS)(cS^{-1}S\tau + d)) \\
= \; & \det(c1_2\tau + d1_2)^{-k} \cdot f(\tilde{M}(S\tau)) \\
= \; & \det(c1_2\tau + d1_2)^{-k} \cdot (f|_k\tilde{M})(S\tau) \cdot \det(c1_2\tau + d1_2)^{k} \\
= \; & (f|_k\tilde{M})(S\tau) \\
= \; & (f|_k\tilde{M})[S](\tau),
\end{aligned}$$

where

$$\tilde{M} = \begin{pmatrix} a1_2 & bS \\ cS^{-1} & d1_2 \end{pmatrix} \in \mathrm{Sp}_2(\mathbb{K}).$$

We can find

$$\gamma \in \mathrm{Sp}_2(\mathcal{O}), \quad R = \begin{pmatrix} \tilde{S} & \tilde{T} \\ 0_2 & \overline{\tilde{S}^{-1}}^T \end{pmatrix} \in \mathrm{Sp}_2(\mathbb{K})$$

such that $\tilde{M} = \gamma R$. We describe the details in lemma 3.11. Then, we have

$$\begin{aligned}
& (f[S]|_{2k}M)(\tau) \\
= \; & (f|_k\tilde{M})[S](\tau) \\
= \; & (f|_k\gamma|_kR)[S](\tau) \\
= \; & (f|_kR)[S](\tau) \\
= \; & \det\left(\overline{\tilde{S}^{-1}}^T\right)^{-k} \cdot f(R(S\tau)) \\
= \; & \overline{\det(\tilde{S})}^k \cdot f\left(\left(\tilde{S} \cdot S\tau + \tilde{T}\right) \cdot \left(\overline{\tilde{S}^{-1}}^T\right)^{-1}\right) \\
= \; & \overline{\det(\tilde{S})}^k \cdot f\left(\tilde{S}S\overline{\tilde{S}}^T\tau + \tilde{T}\overline{\tilde{S}}^T\right).
\end{aligned}$$

Thus, a cusp $\hat{c}$ and $S \in \mathcal{P}_2(\mathcal{O})$ gives us a linear map $f \mapsto f[S]|M_{\hat{c}}$ which we can calculate as described.

Via other methods, we can more directly calculate the vectorspace of Elliptic modular cusp forms in $M_{\hat{c}}$.

By comparing this, we gain new information and we have another method to reduce $\mathcal{M}_i$.

Lemma 3.10. *Let $\hat{c} \in \mathbb{Q} \cup \{\infty\}$. Then there exists a matrix $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathbb{Z})$ such that $\frac{a\infty+b}{c\infty+d} = M\infty = c$.*

Proof. Case 1: $\hat{c} = \infty$. Then we can choose $M = 1_2$.

Case 2: $\hat{c} = 0$. Then we can choose $M = J_2$.

Case 3: $\hat{c} \in \mathbb{Q}^\times$. Let $\hat{c} = \frac{a}{b}$ with $a \in \mathbb{Z}$, $c \in \mathbb{N}$ such that there is no common denominator of a and b. Then we can select $b, d \in \mathbb{Z}$ such that $1 = ac - bd$. Then we have $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathbb{Z})$ and $M\infty = \hat{c}$. ☐

Lemma 3.11 ($solveR$)**.** *Let*

$$\tilde{M} = \begin{pmatrix} a1_2 & bS \\ cS^{-1} & d1_2 \end{pmatrix} \in \mathrm{Sp}_2(\mathcal{O}),$$

where

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}_2(\mathbb{Z}), \quad S \in \mathrm{Her}_2(\mathcal{O}).$$

We can find

$$\gamma \in \mathrm{Sp}_2(\mathcal{O}), \quad R = \begin{pmatrix} \tilde{S} & \tilde{T} \\ 0_2 & \overline{\tilde{S}^{-1}}^T \end{pmatrix} \in \mathrm{Sp}_2(\mathbb{K})$$

such that $\tilde{M} = \gamma R$.

Proof. It follows a lengthy constructive proof.

Consider the embedding of $M_1 = \left(\begin{smallmatrix} a_1 & b_1 \\ c_1 & d_1 \end{smallmatrix}\right), M_2 = \left(\begin{smallmatrix} a_2 & b_2 \\ c_2 & d_2 \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathbb{K})$

$$M_1 \times M_2 := \begin{pmatrix} a_1 & & b_1 & \\ & a_2 & & b_2 \\ c_1 & & d_1 & \\ & c_2 & & d_2 \end{pmatrix} \in \mathrm{Sp}_2(\mathbb{K}).$$

Write

$$\tilde{M}_i = \begin{pmatrix} a_{i,1} & a_{i,2} & b_{i,1} & b_{i,2} \\ a_{i,3} & a_{i,4} & b_{i,3} & b_{i,4} \\ c_{i,1} & c_{i,2} & d_{i,1} & d_{i,2} \\ c_{i,3} & c_{i,4} & d_{i,3} & d_{i,4} \end{pmatrix}$$

for $\tilde{M}_i \in \mathrm{Sp}_2(\mathbb{K})$. We set

$$\tilde{M}_1 := \tilde{M}$$

and

$$\tilde{M}_{i+1} := \gamma_i \tilde{M}_i$$

for $\gamma_i \in \mathrm{Sp}_2(\mathcal{O})$. Note that we have

$$a_{1,1} = a, \qquad a_{1,2} = 0, \qquad\qquad d_{1,1} = d, \qquad d_{1,2} = 0,$$

$$a_{1,3} = 0, \qquad a_{1,4} = a, \qquad\qquad d_{1,3} = 0, \qquad d_{1,4} = d,$$

$$\begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{1,3} & b_{1,4} \end{pmatrix} = bS, \qquad\qquad \begin{pmatrix} c_{1,1} & c_{1,2} \\ c_{1,3} & c_{1,4} \end{pmatrix} = cS^{-1}.$$

1) We want to define γ_1 such that $c_{2,1} = c_{2,3} = 0$. Set

$$\gamma_1 := \begin{pmatrix} a_{\gamma_1,1} & b_{\gamma_1,1} \\ c_{\gamma_1,1} & d_{\gamma_1,1} \end{pmatrix} \times \begin{pmatrix} a_{\gamma_1,4} & b_{\gamma_1,4} \\ c_{\gamma_1,4} & d_{\gamma_1,4} \end{pmatrix}.$$

Given $0 = c_{2,3} = c_{\gamma_1,4} \cdot a_{1,3} + d_{\gamma_1,4} \cdot c_{1,3}$, where $a_{1,3} = 0$, we get $d_{\gamma_1,4} := 0$ and we can set $a_{\gamma_1,4} := 0$, $b_{\gamma_1,4} := 1$ and $c_{\gamma_1,4} := -1$ such that $\left(\begin{smallmatrix} a_{\gamma_1,4} & b_{\gamma_1,4} \\ c_{\gamma_1,4} & d_{\gamma_1,4} \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathcal{O})$.

We want $0 = c_{2,1} = c_{\gamma_1,1} \cdot a_{1,1} + d_{\gamma_1,1} \cdot c_{1,1}$. In case we already have $c_{1,1} = 0$, we can select $\left(\begin{smallmatrix} a_{\gamma_1,1} & b_{\gamma_1,1} \\ c_{\gamma_1,1} & d_{\gamma_1,1} \end{smallmatrix}\right) = 1_2 \in \mathrm{Sp}_1(\mathcal{O})$. Otherwise: We have $c_{1,1} \in \mathbb{K}$ but there is a $l_1 \in \mathcal{O}$ such that $c_{1,1} \cdot l_1 \in \mathcal{O}$. Via *xgcd* (see section 4.6), we can find $d_1, a_{\gamma_1,1}, b_{\gamma_1,1} \in \mathcal{O}$ such that $1 = a_{\gamma_1,1} \frac{a_{1,1} \cdot l_1}{d_1} + b_{\gamma_1,1} \frac{c_{1,1} \cdot l_1}{d_1}$ and $\frac{a_{1,1} \cdot l_1}{d_1}, \frac{c_{1,1} \cdot l_1}{d_1} \in \mathcal{O}$. Then set $c_{\gamma_1,1} := -\frac{c_{1,1} \cdot l_1}{d_1}$ and $d_{\gamma_1,1} := \frac{a_{1,1} \cdot l_1}{d_1}$. Then we have $\left(\begin{smallmatrix} a_{\gamma_1,1} & b_{\gamma_1,1} \\ c_{\gamma_1,1} & d_{\gamma_1,1} \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathcal{O})$.

That way, we have defined $\gamma_1 \in \mathrm{Sp}_2(\mathcal{O})$ and we have $c_{2,1} = c_{2,3} = 0$ in $\tilde{M}_2 = \gamma_1 \tilde{M}$.

2) In case we already have $c_{2,4} = 0$, we can skip this step with $\gamma_2 = 1_4$. Otherwise, we assume $c_{2,4} \neq 0$.

Consider

$$\gamma_2 = \begin{pmatrix} A_{\gamma_2} & 0_2 \\ 0_2 & D_{\gamma_2} \end{pmatrix}$$

where $A_{\gamma_2} \in \mathrm{Sp}_1(\mathcal{O})$ and $D_{\gamma_2} = \overline{A_{\gamma_2}^{-1}}^T \in \mathrm{Sp}_1(\mathcal{O})$. Then we have $\gamma_2 \in \mathrm{Sp}_2(\mathcal{O})$. Write $D_{\gamma_2} = \left(\begin{smallmatrix} d_{\gamma_2,1} & d_{\gamma_2,2} \\ d_{\gamma_2,3} & d_{\gamma_2,4} \end{smallmatrix}\right)$.

Then we have

$$\begin{aligned} \begin{pmatrix} c_{3,1} & c_{3,2} \\ c_{3,3} & c_{3,4} \end{pmatrix} &= D_{\gamma_2} \begin{pmatrix} c_{2,1} & c_{2,2} \\ c_{2,3} & c_{2,4} \end{pmatrix} \\ &= \begin{pmatrix} d_{\gamma_2,1} & d_{\gamma_2,2} \\ d_{\gamma_2,3} & d_{\gamma_2,4} \end{pmatrix} \begin{pmatrix} 0 & c_{2,2} \\ 0 & c_{2,4} \end{pmatrix} \\ &= \begin{pmatrix} 0 & d_{\gamma_2,1} c_{2,2} + d_{\gamma_2,2} c_{2,4} \\ 0 & d_{\gamma_2,3} c_{2,2} + d_{\gamma_2,4} c_{2,4} \end{pmatrix}. \end{aligned}$$

We want $0 = c_{3,4} = d_{\gamma_2,3} \cdot c_{2,2} + d_{\gamma_2,4} \cdot c_{2,4}$. In case we already have $c_{2,2} = 0$, we can select $D_{\gamma_2} := J_1 \in \mathrm{Sp}_1(\mathcal{O})$. Otherwise: Again, $c_{2,2}, c_{2,4} \in \mathbb{K}$ but we can select $l_2 \in \mathcal{O}$ such that $c_{2,2} \cdot l_2, c_{2,4} \cdot l_2 \in \mathcal{O}$. Again via *xgcd*, we can find $d_2, d_{\gamma_2,1}, d_{\gamma_2,2} \in \mathcal{O}$ such that $1 = d_{\gamma_2,1} \frac{c_{2,2} \cdot l_2}{d_2} + d_{\gamma_2,2} \frac{c_{2,4} \cdot l_2}{d_2}$ and $\frac{c_{2,2} \cdot l_2}{d_2}, \frac{c_{2,4} \cdot l_2}{d_2} \in \mathcal{O}$. Set $d_{\gamma_2,3} := -\frac{c_{2,4} \cdot l_2}{d_2}$ and $d_{\gamma_2,4} := \frac{c_{2,2} \cdot l_2}{d_2}$. Then we have $D_{\gamma_2} \in \mathrm{Sp}_1(\mathcal{O})$.

That way, we have defined $\gamma_2 \in \mathrm{Sp}_2(\mathcal{O})$ and we have $c_{3,1} = c_{3,3} = c_{3,4} = 0$ in $\tilde{M}_3 = \gamma_2 \tilde{M}_2 = \gamma_2 \gamma_1 \tilde{M}$.

Note that we have $A_{\gamma_2} = \overline{D_{\gamma_2}^{-1}}^T = \begin{pmatrix} \overline{d_{\gamma_2,4}} & -\overline{d_{\gamma_2,3}} \\ -\overline{d_{\gamma_2,2}} & \overline{d_{\gamma_2,1}} \end{pmatrix}$.

3) Now we

$$\begin{pmatrix} a_{3,1} & a_{3,2} \\ a_{3,3} & a_{3,4} \end{pmatrix} = A_{\gamma_2} \begin{pmatrix} a_{2,1} & a_{2,2} \\ a_{2,3} & a_{2,4} \end{pmatrix},$$

thus

$$a_{3,1} = \overline{d_{\gamma_2,1}} \cdot a_{2,1} - \overline{d_{\gamma_2,3}} \cdot a_{2,3}.$$

We have $\tilde{M}_3 \in \mathrm{Sp}_2(\mathcal{O})$, thus $\overline{\tilde{M}_3}^T \left(\begin{smallmatrix} & -1_2 \\ 1_2 & \end{smallmatrix}\right) \tilde{M}_3 = \left(\begin{smallmatrix} & -1_2 \\ 1_2 & \end{smallmatrix}\right)$. Thus

$$\overline{\begin{pmatrix} a_{3,1} & a_{3,3} & c_{3,1} & c_{3,3} \\ a_{3,2} & a_{3,4} & c_{3,2} & c_{3,4} \\ b_{3,1} & b_{3,3} & d_{3,1} & d_{3,3} \\ b_{3,2} & b_{3,4} & d_{3,2} & d_{3,4} \end{pmatrix}} \cdot \begin{pmatrix} -c_{3,1} & -c_{3,2} & -d_{3,1} & -d_{3,2} \\ -c_{3,3} & -c_{3,4} & -d_{3,3} & -d_{3,4} \\ a_{3,1} & a_{3,2} & b_{3,1} & b_{3,2} \\ a_{3,3} & a_{3,4} & b_{3,3} & b_{3,4} \end{pmatrix} = \begin{pmatrix} & -1_2 \\ 1_2 & \end{pmatrix}.$$

It follows

$$\begin{aligned} & -\overline{a_{3,1}} \cdot c_{3,2} - \overline{a_{3,3}} \cdot c_{3,4} + \overline{c_{3,1}} \cdot a_{3,2} + \overline{c_{3,3}} \cdot a_{3,4} = 0 \\ \overset{c_{3,1}=c_{3,3}=c_{3,4}=0}{\Longrightarrow} \quad & \overline{a_{3,1}} \cdot c_{3,2} = 0 \\ \Longrightarrow \quad & a_{3,1} = 0 \quad \text{or} \quad c_{3,2} = 0. \end{aligned}$$

Thus, either we are ready ($c_{3,2} = 0$) or we are not ($c_{3,2} \neq 0$) but then we have $a_{3,1} = 0$. We need that in the next step.

4) Assume that we have $c_{3,2} \neq 0$ and $a_{3,1} = 0$, otherwise skip this step with $\gamma_3 = 1_4$. Set

$$\gamma_3 := \begin{pmatrix} a_{\gamma_3,1} & b_{\gamma_3,1} \\ c_{\gamma_3,1} & d_{\gamma_3,1} \end{pmatrix} \times \begin{pmatrix} a_{\gamma_3,4} & b_{\gamma_3,4} \\ c_{\gamma_3,4} & d_{\gamma_3,4} \end{pmatrix}.$$

With $\tilde{M}_4 = \gamma_3 \tilde{M}_3$, we have

$$\begin{aligned} c_{4,1} &= c_{\gamma_3,1} \cdot \underbrace{a_{3,1}}_{=0} + d_{\gamma_3,1} \cdot \underbrace{c_{3,1}}_{=0}, \\ c_{4,2} &= c_{\gamma_3,1} \cdot a_{3,2} + d_{\gamma_3,1} \cdot c_{3,2}, \\ c_{4,3} &= c_{\gamma_3,4} \cdot a_{3,2} + d_{\gamma_3,4} \cdot \underbrace{c_{3,2}}_{=0}, \\ c_{4,4} &= c_{\gamma_3,4} \cdot a_{3,4} + d_{\gamma_3,4} \cdot \underbrace{c_{3,4}}_{=0}. \end{aligned}$$

Set $c_{\gamma_3,4} := b_{\gamma_3,4} := 0$ and $a_{\gamma_3,4} := d_{\gamma_3,4} := 1$. Thus $\left(\begin{smallmatrix} a_{\gamma_3,4} & b_{\gamma_3,4} \\ c_{\gamma_3,4} & d_{\gamma_3,4} \end{smallmatrix}\right) = 1_2 \in \mathrm{Sp}_1(\mathcal{O})$. Then we have $c_{4,3} = 0$ and $c_{4,4} = 0$.

Let $l_3 \in \mathcal{O}$ such that $a_{3,2} \cdot l_3, c_{3,2} \cdot l_3 \in \mathcal{O}$. Via $xgcd$, we can find $d_3, a_{\gamma_3,1}, b_{\gamma_3,1} \in \mathcal{O}$ such that $1 = a_{\gamma_3,1} \frac{a_{3,2} \cdot l_3}{d_3} + b_{\gamma_3,1} \frac{c_{3,2} \cdot l_3}{d_3}$ and $\frac{a_{3,2} \cdot l_3}{d_3}, \frac{c_{3,2} \cdot l_3}{d_3} \in \mathcal{O}$. Set $c_{\gamma_3,1} := -\frac{c_{3,2} \cdot l_3}{d_3}$ and $d_{\gamma_3,1} := \frac{a_{3,2} \cdot l_3}{d_3}$. Then we have $\left(\begin{smallmatrix} a_{\gamma_3,1} & b_{\gamma_3,1} \\ c_{\gamma_3,1} & d_{\gamma_3,1} \end{smallmatrix}\right) \in \mathrm{Sp}_1(\mathcal{O})$. And we have $c_{4,2} = 0$.

That way, we have defined $\gamma_3 \in \mathrm{Sp}_2(\mathcal{O})$ and we have $c_{4,1} = c_{4,2} = c_{4,3} = c_{4,4} = 0$ in $\tilde{M}_4 = \gamma_3 \tilde{M}_3 = \gamma_3 \gamma_2 \gamma_1 \tilde{M}$.

5) We are ready. Now, set

$$R := \tilde{M}_4 = \gamma_3\gamma_2\gamma_1\tilde{M} \in \mathrm{Sp}_2(\mathbb{K})$$

and

$$\gamma := \gamma_1^{-1}\gamma_2^{-1}\gamma_3^{-1} \in \mathrm{Sp}_2(\mathcal{O})$$

such that

$$\gamma R = \tilde{M}.$$

Write

$$R = \begin{pmatrix} \tilde{S} & \tilde{T} \\ 0_2 & \tilde{U} \end{pmatrix}.$$

We have $R \in \mathrm{Sp}_2(\mathbb{K})$, thus $J_2[R] = J_2$, i.e.

$$\begin{aligned} & \begin{pmatrix} \overline{\tilde{S}}^T & \\ \overline{\tilde{T}}^T & \overline{\tilde{U}}^T \end{pmatrix} \begin{pmatrix} & -1_2 \\ 1_2 & \end{pmatrix} \begin{pmatrix} \tilde{S} & \tilde{T} \\ & \tilde{U} \end{pmatrix} \\ = & \begin{pmatrix} \overline{\tilde{S}}^T & \\ \overline{\tilde{T}}^T & \overline{\tilde{U}}^T \end{pmatrix} \begin{pmatrix} & -\tilde{U} \\ \tilde{S} & \tilde{T} \end{pmatrix} \\ = & \begin{pmatrix} 0_2 & -\overline{\tilde{S}}^T\tilde{U} \\ \overline{\tilde{U}}^T\tilde{S} & -\overline{\tilde{T}}^T\tilde{U} + \overline{\tilde{U}}^T\tilde{T} \end{pmatrix} \\ = & \begin{pmatrix} & -1_2 \\ 1_2 & \end{pmatrix} \end{aligned}$$

Thus, we have $\tilde{U} = \overline{\tilde{S}^{-1}}^T$. □

Note that this proof gives explicit formulas to calculate γ and R. This has been implemented in `helpers.py` as $solveR$. A list of test cases have also been generated in the function $test\_solveR$ in `tests.py` which also indirectly test the $xgcd$ and $divmod$ implementation (described in section 4.6).

Remark 3.12. We want to develop the formulas to use the new information about the cusp expansion relations as described in preliminaries 3.9. This is very similar as the idea in remark 3.6, such that we can develop similar formulas as in remark 3.8.

Recall that we have

$$(f[S]|_{2k}M)(\tau) = \overline{\det(\tilde{S})}^k \cdot f\left(\tilde{S}S\overline{\tilde{S}}^T\tau + \tilde{T}\overline{\tilde{S}}^T\right)$$

for $\tilde{S}, \tilde{T} \in \mathrm{Sp}_1(\mathbb{K})$ as described in preliminaries 3.9 (via $solveR$ from M and S).

Then, for the Fourier expansion, we have

$$\begin{aligned}(f[S]|_{2k}M)(\tau) &= \overline{\det(\tilde{S})}^k \cdot \sum_{T\in\Lambda} a(T) \cdot e^{2\pi i\cdot \mathrm{tr}\left(T\tilde{S}S\overline{\tilde{S}}^T\tau + T\tilde{T}\overline{\tilde{S}}^T\right)} \\ &= \overline{\det(\tilde{S})}^k \cdot \sum_{T\in\Lambda} a(T) \cdot e^{2\pi i\cdot \mathrm{tr}\left(T\tilde{S}S\overline{\tilde{S}}^T\right)\cdot\tau} \cdot e^{2\pi i\cdot \mathrm{tr}\left(T\tilde{T}\overline{\tilde{S}}^T\right)} \\ &= \sum_{p\in\frac{1}{l}\mathbb{N}_0} (a[S]|_{2k}M)(p)\cdot e^{2\pi i\cdot p\cdot\tau},\end{aligned}$$

where $l \in \mathbb{N}$ such that $\mathrm{tr}\left(T\tilde{S}S\overline{\tilde{S}}^T\right)\cdot l \in \mathbb{N}_0$ for all $T \in \Lambda$ and

$$(a[S]|_{2k}M)(p) = \overline{\det(\tilde{S})}^k \cdot \sum_{\substack{T\in\Lambda,\\ \mathrm{tr}\left(T\tilde{S}S\overline{\tilde{S}}^T\right)=p}} a(T)\cdot e^{2\pi i\cdot\mathrm{tr}\left(T\tilde{T}\overline{\tilde{S}}^T\right)}$$

for all $p \in \frac{1}{l}\mathbb{N}_0$.

Similarily as in remark 3.8, we get a formula to calculate such a matrix for the linear map $a \mapsto a[S]|_{2k}M$. Because we don't just have numbers over $\mathbb{Q}$ and $\mathbb{Z}$ anymore, a performant implementation is a bit more complicated than before. We describe the details in section 4.7.

At this point, let us just remark that we get another superspace $\mathcal{M}^H_{\hat{c},S} \subset \mathbb{Q}^{\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}}$ of $\mathcal{FE}(\mathcal{M}^{\mathbb{H}_2}_k(\Gamma))$. $\square$

3.3 Algorithm

Now we formulate the main algorithm based on the work previously developed in this chapter.

Algorithm 3.13. We have the Hermitian modular form degree $n = 2$ fixed, as well as some Δ (for now, $\Delta \in \{3, 4, 8\}$). Then we select some form weight $k \in \mathbb{Z}$ ($k \in \{1, \ldots, 20\}$ or so), let $\mathcal{O} \subseteq \mathbb{Q}(\sqrt{-\Delta})$ be the maximal order (see section 2.3.1) and let $\Gamma = \mathrm{Sp}_2(\mathcal{O})$. We use $\nu\colon \Gamma \to \mathbb{C}^\times$, $\nu \equiv 1$ as the Abel character (see remark 3.2).

1. Start with $\mathcal{M}_1 := \mathbb{Q}^{\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}}$ and $i = 1$.
2. Enumerate matrices $S \in \mathcal{P}_2(\mathcal{O})$ and for each matrix perform the following steps.
3. In remark 3.6 and more detailed in section 4.7 about the restriction ($f \mapsto f[S]$), we get a superspace $\mathcal{M}_S^H \subset \mathbb{Q}^{\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}}$ of $\mathcal{F}\mathcal{E}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma))$.
4. From remark 3.12 and the cusp expansion information, for each cusp $c \in \mathcal{C}_l \subset \mathbb{Q} \cup \{\infty\}$ with $c \neq \infty$ where $\mathcal{C}_l$ are the cusps in $\Gamma_0(l)$ and $l = \det(S)$, we get a superspace $\mathcal{M}_{c,S}^H \subset \mathbb{Q}^{\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}}$ of $\mathcal{F}\mathcal{E}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma))$.
5. Now set
$$\mathcal{M}_{i+1} := \mathcal{M}_i \cap \mathcal{M}_S^H \cap \bigcap_{c \in \mathcal{C}_l, c \neq \infty} \mathcal{M}_{c,S}^H.$$
6. If
$$\dim \mathcal{M}_i = \dim \mathcal{M}_k^{\mathbb{H}_2}(\Gamma),$$
then we are ready and we have
$$\mathcal{M}_i = \mathcal{F}\mathcal{E}_{\mathcal{F}}(\mathcal{M}_k^{\mathbb{H}_2}(\Gamma)).$$
If not, increase i, return to step 2 and select the next $S \in \mathcal{P}_2(\mathcal{O})$.

Chapter 4

Implementation

In this chapter, we are describing the implementation. All of the code can be found at [Zey13a].

The code consists of several parts. All of it was implemented around the Sage ([S+13]) framework, thus the main language is Python ([vR13]). For performance reasons, some very heavy calculations have been implemented in C++ ([Str83]) and some Cython ([BBS+13]) code is the interface between both parts.

4.1 Basic code structure

4.1.1 Main function $herm\_modform\_space$

The main entry point is in the file `algo.py`. The function $herm\_modform\_space$ calculates the Hermitian modula form space. The function gets the fundamental discriminant $D = -\Delta$, the Hermitian modula forms weight $k = HermWeight$ and the precision limit $B_{\mathcal{F}} = B\_cF$ as its input and returns the vector space of Fourier expansions of Hermitian modular forms to the precision $B_{\mathcal{F}}$. The Fourier expansions are indexed by the reduced matrices of $\mathcal{F}$ (see remark 3.7 for details). This index list can also be returned by $herm\_modform\_indexset$.

The function can also do its calculation in parallel via multiple processes. As a convenience, to easily start the calculation with N processes in parallel, there is the function $herm\_modform\_space\_\_parallel$ with the additional parameter $task\_limit$, where you just set $task\_limit = N$. For details about the parallelization, see section 4.8.

Thus, to calculate the Hermitian modular forms with $D = -3$, weight 6 and $B_{\mathcal{F}} = 7$, you can do:

```
# run sage in the 'src' directory of this work
import algo
algo.herm_modform_space(D=-3, HermWeight=6, B_cF=7)
```

Or, if you want to use 4 processes in parallel:

```
algo.herm_modform_space__parallel(
    D=-3, HermWeight=6, B_cF=7, task_limit=4)
```

The function $herm\_modform\_space$ uses $modform\_restriction\_info$ and $modform\_cusp\_info$ which are also defined in the same file. The theory behind these functions is described in section 3.1 and section 3.2, accordingly.

Both return a vector space which is a superspace of the Hermitian modular form Fourier expansions and $herm\_modform\_space$ intersects them until the final dimension is reached.

In the rest of this chapter, we will demonstrate the details of the calculations and representations.

4.2 $\mathcal{O}$ and $\mathcal{O}^{\#}$ representation and calculations

To represent $\mathcal{O}$ and $\mathcal{O}^{\#}$ in code, mostly in the low level C++ code (files `algo_cpp.cpp`, `structs.hpp`, `reduceGL.hpp`), we can use two integers in both cases as the coefficients of some basis.

Most of the calculations presented in this section are implemented in `structs.hpp`.

4.2.1 Representations

For $a \in \mathcal{O}$, we use

$$a = a_1 + a_2 \frac{D + \sqrt{D}}{2}$$

with $a_1, a_2 \in \mathbb{Z}$. It holds

$$\begin{aligned}
\Re(a) &= a_1 + a_2 \frac{D}{2}, \\
\Re(a)^2 &= a_1^2 + D a_1 a_2 + \frac{D^2}{4} a_2^2, \\
\Im(a) &= a_2 \frac{\sqrt{-D}}{2}, \\
\Im(a)^2 &= a_2^2 \frac{-D}{4}, \\
|a|^2 &= \Re(a)^2 + \Im(a)^2 = a_1^2 - (-D) a_1 a_2 + \frac{D^2 - D}{4} a_2^2.
\end{aligned}$$

Note that 4 divides $D^2 - D$. Thus, $|a|^2 \in \mathbb{Z}$.

Sometimes we have given $a \in \mathbb{K}$ where we easily have $\Re(a)$ and $\Im(a)$ available and we want to calculate $a_1, a_2 \in \mathbb{Q}$ in the above representation. We get

$$\begin{aligned} a_2 &= \Im(a) \frac{2}{\sqrt{-D}}, \\ a_1 &= \Re(a) - a_2 \frac{D}{2} = \Re(a) + \Im(a)\sqrt{-D}. \end{aligned}$$

For $b \in \mathcal{O}^{\#}$, we use

$$b = b_1 \frac{1}{\sqrt{D}} + b_2 \frac{1+\sqrt{D}}{2}$$

with $b_1, b_2 \in \mathbb{Z}$. It holds

$$\begin{aligned} \Re(b) &= \frac{1}{2} b_2, \\ \Re(b)^2 &= \frac{1}{4} b_2^2, \\ \Im(b) &= -\frac{b_1}{\sqrt{-D}} + \frac{1}{2}\sqrt{-D} b_2, \\ \Im(b)^2 &= \frac{b_1^2}{-D} - b_1 b_2 + \frac{1}{4}(-D) b_2^2, \\ |b|^2 = \Re(b)^2 + \Im(b)^2 &= \frac{b_1^2}{-D} - b_1 b_2 + \frac{1}{4}(1-D) b_2^2. \end{aligned}$$

When we need $|b|^2$ in an implementation, we can multiply it with $-D$ to get an integer:

$$(-D)|b|^2 = b_1^2 - (-D) b_1 b_2 + \frac{D^2 - D}{4} b_2^2.$$

When we have $b \in \mathbb{K}$ where $\Re(b)$ and $\Im(b)$ are easily available and when we want to calculate $b_1, b_2 \in \mathbb{Q}$ in the above representation, we get

$$\begin{aligned} b_2 &= 2\Re(b), \\ b_1 &= b_2 \frac{-D}{2} - \Im(b)\sqrt{-D} = \Re(b)(-D) - \Im(b)\sqrt{-D}. \end{aligned}$$

Let us calculate the complex conjugate $\overline{b}$ of $b \in \mathcal{O}^{\#}$:

$$\begin{aligned} \overline{b} &= \frac{-b_1}{\sqrt{D}} + \frac{b_2}{2} - b_2 \frac{\sqrt{D}}{2} \\ &\overset{!}{=} \hat{b}_1 \frac{1}{\sqrt{D}} + \hat{b}_2 \frac{1+\sqrt{D}}{2} \\ \Rightarrow \quad \hat{b}_2 &= b_2, \\ \hat{b}_1 &= \overline{b}\sqrt{D} - \hat{b}_2(\sqrt{D} + D)\tfrac{1}{2} \\ &= b_2 \frac{\sqrt{D}}{2} - b_2 \frac{\sqrt{D}}{2} - b_2 \frac{D}{2} - b_2 \frac{D}{2} - b_1 \\ &= -b_2 D - b_1. \end{aligned}$$

Note that $b \in \mathbb{R}$ if and only if $b_1 \frac{1}{\sqrt{D}} = -b_2 \frac{\sqrt{D}}{2}$, i.e.

$$2b_1 = -b_2 D.$$

4.2.2 Multiplications

Let $a, b \in \mathcal{O}$ with $a = a_1 + a_2 \frac{D+\sqrt{D}}{2}$, $b = b_1 + b_2 \frac{D+\sqrt{D}}{2}$. Then we have

$$\begin{aligned} a \cdot b &= a_1 b_1 + a_1 b_2 (D + \sqrt{D}) \tfrac{1}{2} + b_1 a_2 (D + \sqrt{D}) \tfrac{1}{2} + a_2 b_2 \tfrac{1}{4} \underbrace{(D^2 + 2D\sqrt{D} + D)}_{=2D(D+\sqrt{D})-D^2+D} \\ &= \frac{\sqrt{D} + D}{2}(a_1 b_2 + b_1 a_2 + D a_2 b_2) + a_1 b_1 - a_2 b_2 \frac{D^2 - D}{4}. \end{aligned}$$

Now, let $a \in \mathcal{O}^{\#}$ and $b \in \mathcal{O}$ with

$$\begin{aligned} a &= a_1 \frac{1}{\sqrt{D}} + a_2 \frac{1 + \sqrt{D}}{2}, \\ b &= b_1 + b_2 \frac{D + \sqrt{D}}{2}. \end{aligned}$$

Then we have

$$\begin{aligned} a \cdot b &= a_1 b_1 \tfrac{1}{\sqrt{D}} + a_1 b_2 (\sqrt{D} + 1) \tfrac{1}{2} + a_2 b_1 (1 + \sqrt{D}) \tfrac{1}{2} + a_2 b_2 \underbrace{(D + \sqrt{D} + D\sqrt{D} + D)}_{\substack{= 2D + \sqrt{D} + D\sqrt{D} \\ = 2D + \sqrt{D}(1 + D)}} \tfrac{1}{4} \\ &= a_1 b_1 \tfrac{1}{\sqrt{D}} + (a_1 b_2 + a_2 b_1)(1 + \sqrt{D}) \tfrac{1}{2} + a_2 b_2 (2D + \sqrt{D}(1 + D)) \tfrac{1}{4}. \end{aligned}$$

Thus, when representing $a \cdot b \in \mathcal{O}^{\#}$ as

$$a \cdot b = (ab)_1 \frac{1}{\sqrt{D}} + (ab)_2 \frac{1 + \sqrt{D}}{2},$$

we get

$$(ab)_2 = a_1 b_2 + a_2 b_1 + a_2 b_2 D$$

and

$$\begin{aligned}(ab)_1 &= \sqrt{D}ab - (ab)_2(\sqrt{D}+D)\tfrac{1}{2}\\ &= a_1b_1 + (a_1b_2+b_1a_2)(\sqrt{D}+D)\tfrac{1}{2} + a_2b_2(D+\sqrt{D})^2\tfrac{1}{4}\\ &\quad - (a_1b_2+a_2b_1+a_2b_2D)(\sqrt{D}+D)\tfrac{1}{2}\\ &= a_1b_1 + a_2b_2\underbrace{\left((D+\sqrt{D})^2\tfrac{1}{4} - D(\sqrt{D}+D)\tfrac{1}{2}\right)}_{\substack{=\frac{D^2}{4}+\frac{D\sqrt{D}}{2}+\frac{D}{4}-\frac{D\sqrt{D}}{2}-\frac{D^2}{2}\\ =\frac{D^2-D}{4}}}\\ &= a_1b_1 + a_2b_2\frac{D^2-D}{4}.\end{aligned}$$

4.2.3 Determinant of 2-by-2 matrices

For $[a,b,c] \in \mathrm{Her}_2(\mathbb{C})$, we have

$$\det([a,b,c]) = ac - b\bar{b} = ac - |b|^2.$$

When we have $b \in \mathcal{O}$ or $b \in \mathcal{O}^\#$, we have given a formula for $|b|^2$ in section 4.2.1. With those representations and $a,c \in \mathbb{Z}$, for $b \in \mathcal{O}$, we have

$$\det([a,b,c]) = ac - b_1^2 + (-D)b_1b_2 - \tfrac{D^2-D}{4}b_2^2 \in \mathbb{Z}$$

and for $b \in \mathcal{O}^\#$, we have

$$\det([a,b,c]) = ac - b_1^2\tfrac{1}{-D} + b_1b_2 - \tfrac{1}{4}(1-D)b_2^2 \in \tfrac{1}{-D}\mathbb{Z}.$$

In the code, we represent both matrices $\mathrm{Her}_2(\mathcal{O})$ and $\mathrm{Her}_2(\mathcal{O}^\#)$ by 4-tuples $(a,b_1,b_2,c) \in \mathbb{Z}^4$.

4.2.4 Trace of TS

We want to calculate tr(TS) for $T \in \mathrm{Her}_2(\mathcal{O}^\#)$, $S \in \mathrm{Her}_2(\mathcal{O})$. Let $T = [T_a, T_b, T_c]$ and $S = [S_a, S_b, S_c]$ with

$$\begin{aligned}T_b &= T_{b1}\frac{1}{\sqrt{D}} + T_{b2}\frac{1+\sqrt{D}}{2},\\ S_b &= S_{b1} + S_{b2}\frac{D+\sqrt{D}}{2}\end{aligned}$$

and we have

$$\overline{S_b} = S_{b1} + S_{b2}\frac{D-\sqrt{D}}{2}.$$

Then,

$$\text{tr}(TS) = T_a S_a + \underbrace{T_b \overline{S_b} + \overline{T_b} S_b}_{=2\Re(T_b \overline{S_b})} + T_c S_c$$

and

$$\begin{aligned}\overline{S_b} T_b &= S_{b1} T_{b1} \tfrac{1}{\sqrt{D}} + S_{b1} T_{b2} (1 + \sqrt{D}) \tfrac{1}{2} + S_{b2} D \tfrac{1}{2} T_{b1} \tfrac{1}{\sqrt{D}} - S_{b2} \tfrac{1}{2} T_{b1} \\ &\quad + T_{b2} S_{b2} \tfrac{1}{4} \underbrace{(D - \sqrt{D} + D\sqrt{D} - D)}_{=\sqrt{D}(D-1)} \\ \Rightarrow \Re(\overline{S_b} T_b) &= S_{b1} T_{b2} \tfrac{1}{2} - S_{b2} T_{b1} \tfrac{1}{2}.\end{aligned}$$

Thus, in our Computer implementation, we can just use

$$\text{tr}(TS) = T_a S_a + T_c S_c + S_{b1} T_{b2} - S_{b2} T_{b1}.$$

And if we have $T_a, T_{b1}, T_{b2}, T_c, S_a, S_{b1}, S_{b2}, S_c \in \mathbb{Z}$, we also have $\text{tr}(TS) \in \mathbb{Z}$.

4.3 Iteration of the precision Fourier indice $\mathcal{F}$

The set $\mathcal{F}$ depends on a limit $B_{\mathcal{F}} \in \mathbb{N}$:

$$\mathcal{F} = \mathcal{F}_B = \left\{ \begin{pmatrix} a & b \\ \overline{b} & c \end{pmatrix} \in \Lambda \;\middle|\; 0 \le a, c < B_{\mathcal{F}} \right\} \subseteq \Lambda.$$

In remark 3.7, we see that $\mathcal{F}$ is finite.

We have implemented an iteration of $\mathcal{F}$ in a way that the list of $\mathcal{F}_{B_2}$ always starts with $\mathcal{F}_{B_1}$ if $B_1 \le B_2$. That is *PrecisionF* in `algo_cpp.cpp`. For testing and demonstration purpose, there is also a pure Python implementation *curlF_iter_py* in `helpers.py`. I.e., in Python, for some D and $B1 \le B2$, it yields:

```
curlF1 = list(curlF_iter_py(D=D, B_cF=B1))
curlF2 = list(curlF_iter_py(D=D, B_cF=B2))
assert curlF1 == curlF2[:len(curlF1)]
```

The algorithm of the iteration of $T \in \mathcal{F}$ works in the following way: We have the current matrix represented as integers $a, b_1, b_2, c \in \mathbb{Z}$ and we start with each of them set to 0. Then, $b = b_1 \frac{1}{\sqrt{D}} + b_2 \frac{1+\sqrt{D}}{2}$ and $T = [a, b, c]$. We have the limit $B_{\mathcal{F}} \in \mathbb{N}_0$ and iterate an internal limit $\tilde{B} \in \{0, 1, \ldots, B_{\mathcal{F}} - 1\}$.

1. If the current saved matrix is a valid one, i.e. its determinant is not negative and $0 \le a, c \le B_{\mathcal{F}}$, we return it.

2. We iterate b_2 through $\{0, 1, -1, 2, -2, \ldots\}$.

3. The absolut limit for b_2 is given by

$$4ac \geq b_2^2.$$

Proof. With $\det(T) \geq 0$, we have (see section 4.2.3)

$$(-D)ac \geq b_1^2 - (-D)b_1b_2 + \frac{(-D)(1-D)}{4}b_2^2.$$

And it yields

$$b_1^2 - (-D)b_1b_2 = (b_1 - \tfrac{-D}{2}b_2)^2 - \tfrac{D^2}{4}b_2^2 \geq -\tfrac{D^2}{4}b_2^2,$$

thus

$$(-D)ac \geq -\tfrac{D^2}{4}b_2^2 + \frac{(-D)(1-D)}{4}b_2^2 = \tfrac{-D}{4}b_2^2.$$

This is equivalent with the inequality to-be-proved. $\square$

4. Once we hit that limit, we reset $b_2 := 0$ and we do one iteration step for b_1 through the set $\{0, 1, -1, 2, -2, \ldots\}$.

5. The absolut limit for b_1 is given by

$$ac(D^2 - D) \geq b_1^2.$$

Proof. We have

$$\tfrac{D^2-D}{4}b_2^2 - (-D)b_1b_2 = \left(\sqrt{\tfrac{D^2-D}{4}}b_2 - \frac{-D}{2\sqrt{\tfrac{D^2-D}{4}}}b_1\right)^2 - \tfrac{(-D)^2}{D^2-D}b_1^2 \geq -\tfrac{D^2}{D^2-D}b_1^2.$$

Then, again with $\det(T) \geq 0$ like in the limit for b_2, we have

$$(-D)ac \geq b_1^2 - \frac{D^2}{D^2-D}b_1^2 = b_1^2\frac{-D}{D^2-D}.$$

This is equivalent with the inequality to-be-proved. $\square$

6. Once we hit that limit, we reset $b_1 := b_2 := 0$ and we increase c by one.

7. Once we hit $c > \tilde{B}$, we reset $b_1 := b_2 := 0$ and we increase a by one, if $a < \tilde{B}$. For all cases where $a < \tilde{B}$, we set $c := \tilde{B}$, otherwise $c := 0$.

8. Once we hit $a = \tilde{B}$, we increase $\tilde{B}$ by one and reset $a := 0$ and $c := \tilde{B}$.

9. Once we hit $\tilde{B} \geq B_{\mathcal{F}}$, we are finished.

We have seen in remark 3.7 that it is sufficient to use $\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}$ as the index set. In our implementation, we iterate through $\mathcal{F}$ and save the first occurrence of a new reduced matrix in a list. That list is returned by the function *herm_modform_indexset* which is declared in `helpers.py`. It uses the C++ implementation in `algo_cpp.cpp` as its backend. For testing and demonstration purpose, there is also a pure Python implementation *herm_modform_indexset_py* in `helpers.py`.

4.4 Iteration of $S \in \mathcal{P}_2(\mathcal{O})$

The matrices $S \in \mathcal{P}_2(\mathcal{O})$ are used for the restriction in $f[S]$ for an Hermitian modula form f as described in section 3.1.

There are multiple implementations of this infinite iteration. Our first version only iterated through reduced matrices $\mathcal{P}_2(\mathbb{Z})$ with increasing determinator. We want the increasing determinant because we want to exhaust all possible matrices with low determinants because they are easier for the rest of the calculations. Later, it turned out that matrices only over $\mathbb{Z}$ don't yield enough information and we need matrices with imaginary components. Once you add the imaginary component, it is not possible anymore to iterate through all of $\mathcal{P}_2(\mathbb{Z})$ with increasing determinant because there can be infinity many matrices for a given determinant (or it is not trivial to see if there are not and how to set the limits in an implementation). Thus, the second implementation for matrices over $\mathcal{P}_2(\mathcal{O})$ does not keep the determinant fixed and rather works very similar to the iteration through $\mathcal{F}$, as described in section 4.3.

The implementation is in C++ in the file `algo_cpp.cpp`. The class *CurlS_Generator* owns and manages the iterator and can store several matrices at once because the main matrix calculation implementations (section 4.7 and section 4.7) can be done for several matrices at once. The class *M2T_O_PosDefSortedZZ_Iterator* implements the iteration through $\mathcal{P}_2(\mathbb{Z})$ with increasing denominator. The class *M2T_O_PosDefSortedGeneric_Iterator* implements the generic iteratoin through $\mathcal{P}_2(\mathcal{O})$.

The infinite iteration through $S \in \mathcal{P}_2(\mathbb{Z})$ in *M2T_O_PosDefSortedZZ_Iterator* works as follows: We represent S as $a, b, c \in \mathbb{Z}$ with $[a, b, c] = S$. We start with each of them set to zero. Also, we internally save the current determinant δ and start with $\delta := 0$.

1. We return a matrix if it is valid and reduced. That means that we only return if $a \leq c$, $\det([a, b, c]) = ac - b^2 = \delta$ and if there is no common divisor of a, b, c except 1.
2. We increase c by one. We set $a := \lfloor \frac{\delta + b^2}{c} \rfloor$.

3. Once we hit $c > \delta + b^2$, we reset $c := 0$ and make one iteration step for $b \in \{0, 1, -1, 2, -2, \ldots\}$.
4. Once we hit $3b^2 > \delta$, we know that there aren't any further matrices with this determinant δ. Thus we reset $a := b := c := 0$ and increase δ by one.

Proof. For a reduced matrix $[a, b, c]$, we have

$$0 \leq 2|b| \leq a \leq c.$$

Thus,

$$\delta = ac - b^2 \geq (2|b|)(2|b|) - b^2 = 3b^2. \qquad \square$$

The infinite iteration through $S \in \mathcal{P}_2(\mathcal{O})$ in $M2T\_O\_PosDefSortedGeneric\_Iterator$ is mostly the same as the iteration of $\mathcal{F}$ as described in section 4.3. The difference is that $\mathcal{F}$ is over $\mathcal{O}^\#$ and S is over $\mathcal{O}$. This yields to other limits for b_1 and b_2. Also, we don't have a limit like $\tilde{B}$.

1. We iterate b_2 through $\{0, 1, -1, 2, -2, \ldots\}$.
2. Once we hit the absolut limit of b_2, we reset $b_2 := 0$ and make one iteration step for $b_1 \in \{0, 1, -1, 2, -2, \ldots\}$.
3. Once we hit the absolut limit of b_1, we reset $b_1 := b_2 := 0$ and increase c by one.
4. Once we hit $c > a$, we reset $c := b_1 := b_2 := 0$ and increase a by one.

The absolut limit of b_2 is given by

$$4ac \geq (-D)b_2^2$$

and the absolut limit of b_1 is given by

$$ac(1 - D) \geq b_1^2.$$

Proof. We have $\det(S) \geq 0$ and thus (see section 4.2.3)

$$ac \geq b_1^2 - (-D)b_1 b_2 + \frac{D^2 - D}{4} b_2^2.$$

Note that this is mostly like the inequality in the case over $\mathcal{O}^\#$, except that we have ac on the left side instead of $(-D)ac$. Thus, we can mostly reuse the b_1, b_2 limit calculations from section 4.3. For b_1, we have

$$ac \geq b_1^2 \frac{-D}{D^2 - D}.$$

This is equivalent to the inequality to-be-proved. And for b_2, we have

$$ac \geq \tfrac{-D}{4} b_2^2. \qquad \square$$

4.5 $reduceGL$

In remark 3.7, we have described that it is sufficient to use reduced matrices $\hat{T} \in \mathcal{F}$. Thus, in our implementation, for a given matrix $T \in \mathcal{F}$, we need a way to calculate the reduced matrix $\hat{T} \in \mathcal{F}$ such that

$$\hat{T}[U_T] = T$$

for some $U_T \in \mathrm{GL}_2(\mathcal{O})$. In the code, we don't need U_T directly but rather the determinant of U_T.

Dominic Gehre and Martin Raum have developed a Cython implementation [GR09] of "Functions for reduction of fourier indice of Hermitian modular forms". This function $reduceGL$ gets a matrix $T \in \mathrm{Her}_2(\mathcal{O}^\#)$ and returns the Minkowski-reduced matrix $\hat{T} \in \mathrm{Her}_2(\mathcal{O}^\#)$ and some character evaluation of U_T which also declares the determinant of U_T.

In this work, this function $reduceGL$ has been reimplemented in C++ (`reduceGL.hpp`) and in Python (`reduceGL.py`).

4.6 $divmod$ and $xgcd$

We have given numbers $a, b \in \mathcal{O}$ and we search for $d, p, q \in \mathcal{O}$ such that $d = pa + qb$ and d divides a and b. Then, d is also the greatest common divisor (gcd). This is also equivalent to

$$1 = p\frac{a}{d} + q\frac{b}{d}.$$

For example, we need that in $solveR$ (lemma 3.11).

The extended Euclidean algorithm ($xgcd$) is the standard algorithm to calculate these numbers. It works over all Euclidean domains. In our case, it works for $\Delta \in \{1, 2, 3, 7, 11\}$.

Sage has $xgcd$ which works only for integers. It doesn't directly offer functions to calculate the $xgcd$ over quadratic imaginary number fields.

Thus, in this work, we have reimplemented a simple canonical version of $xgcd$ for $\mathcal{O}$ with a few fast paths, e.g. in the case of integers. This implementation can be found in the class *CurlO* in `helpers.py`.

The main work is done in the *divmod* function. *divmod* gets two numbers $a, b \in \mathcal{O}$ and returns $q, r \in \mathcal{O}$ such that $qb + r = a$. This is the division with remainder. It holds that $f(r) < f(b)$ for the Euclidean Norm $f\colon \mathbb{K} \to \mathbb{R}_{\geq 0}$. In our case, we have $f(x) = |x|$. The current implementation of *divmod* is very naive and should be improved. It can be found as well in the class *CurlO* in `helpers.py`.

Let $a, b \in \mathcal{O}$ with $q = \frac{a}{b}$ be represented in the base $(1, \frac{D+\sqrt{D}}{2})$ as described in section 4.2.1 as tuples $a_1, a_2, b_1, b_2 \in \mathbb{Z}$ and $q_1, q_2 \in \mathbb{Q}$. We can describe the equation $bq = a$ as a matrix multiplication

$$\tilde{B} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}.$$

With the inverse (if it exists), we can calculate q:

$$\tilde{B}^{-1} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}.$$

The multiplation formulas for $\mathcal{O}$ as described in section 4.2.2 yield

$$\tilde{B} = \begin{pmatrix} b_1 & -b_2 \frac{D^2-D}{4} \\ b_2 & b_1 + b_2 D \end{pmatrix}.$$

Then, for the inverse, we have

$$\tilde{B}^{-1} = \frac{1}{\det(\tilde{B})} \begin{pmatrix} b_1 + b_2 D & b_2 \frac{D^2-D}{4} \\ -b_2 & b_1 \end{pmatrix}.$$

For the determinant, we have

$$\begin{aligned} \det(\tilde{B}) &= b_1^2 + b_1 b_2 D + b_2^2 \frac{D^2 - D}{4} \\ &= (b_1 + b_2 \tfrac{D}{2})^2 \underbrace{-b_2^2 \tfrac{D^2}{4} + b_2^2 \tfrac{D^2-D}{4}}_{=-b_2^2 \frac{D}{4} \geq 0} \geq 0. \end{aligned}$$

Note that $\det(\tilde{B}) = 0$ exactly if and only if $b_1 = b_2 = 0$, as it was expected.

This gives us some direct formulas for $q_1, q_2 \in \mathbb{Q}$ which are used in the *divmod* implementation where we select $q_1', q_2' \in \mathbb{Z}$ close to q_1, q_2 such that $r := a - q'b$ becomes minimal with regards to the Euclidean Norm.

4.7 Calculating the matrix of the map $a \to a[S]$

In lemma 3.1, we have seen that, via a matrix $S \in \mathcal{P}_2(\mathcal{O})$, we can restrict a Hermitian modula form $f : \mathbb{H}_2 \to \mathbb{C}$ to an Elliptic modular form $f[S]$. In the whole section 3.1, we have developed the theory.

Let $N = \#\left(\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})}\right)$ with $\mathcal{F}^{\mathrm{GL}_2(\mathcal{O})} = \{T_1, \ldots, T_N\}$ and let $M \in \mathrm{Mat}_{\mathcal{F}(S)\times N}(\mathbb{Q})$ be the matrix to the linear map of Fourier expansions $a \in \mathbb{Q}^{\mathcal{F}}$ of Hermitian modular forms to Fourier expansions $a[S] \in \mathbb{Q}^{\mathcal{F}(S)}$ of Elliptic modular forms. In remark 3.8, we have given the necessary formula

$$M_{i,j} = \sum_{T\in\mathcal{F},\mathrm{tr}(ST)=i,j_T=j} \det(U_T)^{-k}$$

for the i-th row and j-th column with $0 \le i < \mathcal{F}(S)$ and $1 \le j \le N$, where k is the weight of the Hermitian modular forms. j_T is uniquely determined such that the reduced matrix of T is T_{j_T} and $T_{j_T}[U_T] = T$.

In any case, we need to iterate through all of $\mathcal{F}$ to get all the summands in the every sum of every matrix entry. Such an iteration is described in section 4.3. $\mathcal{F}$ is quite huge (see remark 3.7; e.g. for $D = -3$, $B = 10$, we have $\#\mathcal{F} = 21892$) and we need to calculate the reduced matrix index j_T and U_T from each T via $reduceGL$ (see section 4.5) which is heavy to calculate, thus we want to call $reduceGL$ only once for each $T \in \mathcal{F}$. The calculation is still heavy, thus it was implemented in C++ for maximal performance. The algorithm works as follows:

1. *Input*: The restriction matrix $S \in \mathcal{P}_2(\mathcal{O})$, as well as the parameters for the Hermitian modular forms, i.e. the fundamental discriminant D for the underlying quadratic imaginary field, the weight k and the precision limit $B_{\mathcal{F}}$.

   Output: $M \in \mathrm{Mat}_{\mathcal{F}(S)\times N}(\mathbb{Q})$.
2. The outer loop goes through all $T \in \mathcal{F}$ (see section 4.3).
3. For each T, we call $reduceGL$ (see section 4.5) to calculate T_{j_T} and $\det(U_T)$. This gives us also the matrix column $j := j_T$.
4. We also calculate $i := \mathrm{tr}(ST)$ to get the matrix row. In section 4.2.4, we have shown the formula for the direct calculation and also that we have only entries in $\mathbb{N}_0$. We could get $i \ge \mathcal{F}(S)$ but we ignore those.
5. We increase the matrix entry (i, j) by $\det(U_T)^{-k}$.

This algorithm has been implemented in the function $calcMatrix$ in the class $ReductionMatrices\_Calc$ in the file `algo_cpp.cpp`. All the state and parameters are stored in the class so that we need to copy as less data as possible for successive $S \in \mathcal{P}_2(\mathcal{O})$. That is also why the iteration through different $S \in \mathcal{P}_2(\mathcal{O})$ (see section 4.4) has been done in C++.

In Python, in $modform\_restriction\_info$ in the file `algo.py`, we get an instance of that C++ class and call the function $calcMatrix$. This gives us the matrix M_S. Define its column module as

$$\mathcal{M}_S := \left\{ M_S a \,\middle|\, a \in \mathbb{Q}^{\mathcal{F}} \right\}.$$

Via other methods in Sage, we can calculate the vector space $\mathcal{FE}_{\mathcal{F}(S)}(\mathcal{M}_k(\Gamma_0(l_S)))$ of Fourier expansions of Elliptic modular forms to $\Gamma_0(l_S)$ where $l_S := det(S)$ and weight $2k$. Then consider the intersection

$$\mathcal{M}'_S := \mathcal{FE}_{\mathcal{F}(S)}(\mathcal{M}_{2k}(\Gamma_0(l)) \cap \mathcal{M}_S.$$

Now, take them back to the Hermitian modular form space:

$$\mathcal{M}^H_S := \left\{ a \in \mathbb{Q}^{\mathcal{F}} \;\middle|\; Ma \in \mathcal{M}'_S \right\}.$$

In Sage, we can do that by using $solve\_right$ on the matrix M_S and adding the right kernel of M_S.

For testing and demonstration purpose, another implementation has been done in Python in $calcRestrictMatrix\_py$ in the file `helpers.py`. Also for testing purpose, the C++ version can be called directly via the Python function $calcRestrictMatrix\_any$.

4.8 Parallelization

Chapter 5

Conclusion and further work

Blub

Chapter 6

References

[BBS+13] R. Bradshaw, S. Behnel, D. S. Seljebotn, G. Ewing, et al. *The Cython compiler*, 2013. `http://cython.org`.

[Der01] T. Dern. *Hermitesche Modulformen zweiten Grades*. Mainz, 2001.

[GR09] D. Gehre and M. Raum. *Functions for reduction of fourier indice of Hermitian modular forms*, 2009. `https://github.com/martinra/psage/blob/master/psage/modform/hermitianmodularforms/hermitianmodularformd2_fourierexpansion_cython.pyx`.

[PY07] C. Poor and D.S. Yuen. Computations of spaces of siegel modular cusp forms. *Journal of the Mathematical Society of Japan*, 59(1):185–222, 2007.

[Rau12] M. Raum. Computing Jacobi Forms and Linear Equivalences of Special Divisors. *ArXiv e-prints*, December 2012.

[S+13] W. A. Stein et al. *Sage Mathematics Software (Version 5.9)*. The Sage Development Team, 2013. `http://www.sagemath.org`.

[Str83] B. Stroustrup. *The C++ programming language*, 1983.

[vR13] G. van Rossum. *The Python programming language*, 2013. `http://www.python.org`.

[Zey13a] A. Zeyer. *Code repository for this work on GitHub*, 2013. `https://github.com/albertz/diplom-thesis-math/`.

[Zey13b] A. Zeyer. *My homepage*, 2013. `http://www.az2000.de`.