Explicit Representation Results for Quadratic Forms over $\mathbb{Q}$ and $\mathbb{Q}(\sqrt{5})$ by Analytic Methods

by

Katherine Elizabeth Thompson

(Under the direction of Daniel Krashen)

Abstract

In this thesis, we examine representation of positive integers by certain definite quaternary quadratic forms Q over $\mathbb{Z}$ and $\mathbb{Z}[(1+\sqrt{5})/2]$ by studying the theta series of the forms, which are (Hilbert) modular forms with weight 2 and of level and character determined by Q. Specifically, we write the theta series for the following three quadratic forms as the sum of an explicit Eisenstein series and linear combination of normalized Hecke eigencusp forms: $Q_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2$ over $\mathbb{Z}$ (which has long been studied, and which we provide for the sake of introduction), $Q_2(\vec{x}) = x_1^2+x_2^2+x_3^2+7x_4^2$ over $\mathbb{Z}$ (which seems not to have appeared this explicitly previously in the literature), and of Q_1 over $\mathbb{Z}[(1+\sqrt{5})/2]$. We also provide an explicit formula for the Eisenstein series Fourier coefficients appearing in the theta series associated to $Q_3(\vec{x}) = x_1^2+x_2^2+3x_3^2+3(3+\sqrt{5})x_4^2$ over $\mathbb{Z}[(1+\sqrt{5})/2]$. Beyond representation results, we develop and implement an algorithm which counts the number of representations of totally positive integers by a quaternary definite integral quadratic form Q defined over the ring of integers of any real quadratic number field. We also implement an algorithm of Dembélé which returns a Hecke eigenbasis for spaces of Hilbert modular cusp forms of parallel weight two and trivial character via an explicit version of the Eichler-Shimizu-Jacquet-Langlands correspondence.

A primary tool for studying the theta series is Siegel's product formula to compute the Eisenstein components of the Fourier coefficients. To understand the cuspidal component of the theta series, we first find a Hecke eigenbasis for the corresponding modular cusp space and write the cuspidal component of the theta series as a linear combination of these basis elements. Then, using known asymptotics on the Eisenstein coefficients and bounds by Deligne on the cusp form coefficients we prove that all sufficiently large locally represented integers are represented. In particular for the form Q_2, we show that all locally represented $m \geq 15825810$ are represented; we also present for this form upper and lower bounds on $r_Q(m)$ explicitly obtained from the theta series decomposition.

INDEX WORDS: Number theory, Quadratic forms, Modular forms, Quaternion algebras

Explicit Representation Results for Quadratic Forms over $\mathbb{Q}$ and $\mathbb{Q}(\sqrt{5})$ by Analytic Methods

by

Katherine Elizabeth Thompson

B.S., Oglethorpe University, 2007

M.A., The University of Georgia, 2009

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2014

Explicit Representation Results for Quadratic Forms over $\mathbb{Q}$ and $\mathbb{Q}(\sqrt{5})$ by Analytic Methods

by

Katherine Elizabeth Thompson

Major Professor:	Daniel Krashen
Committee:	Edward Azoff
	Pete L. Clark
	Jonathan Hanke

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2014

Dedication

To Betty Lee Hinson (1932-2001): thank you for making it possible for me to learn the geography you never thought you'd need.

Acknowledgments

Thanks to the National Science Foundation for supporting years of my progress and providing me with a means to travel to conferences and to purchase much-needed texts. Along that vein, I would like to thank the graduate committee of the math department at the University of Georgia for their financial and educational support.

The secretaries of the math department–Laura Ackerley, Tamara Higgs, Christie MacDonald, Julie McEver, Connie Stevens, and Gail Suggs–deserve a special thanks. Without them, our world would be frightening, displeasing, and chaotic. They deserve acknowledgement in every paper, book, and thesis the department produces.

I am grateful to the following friends and colleagues for their contribution towards my mathematical career, my development as a teacher, and/or my maintenance of sanity: Ed Azoff, Sybilla Beckmann, Pete Clark, Cheryl Davis, Lassina Dembélé, Jennifer Ellis, Xander Faber, Angela Gibney, Jacob Hicks, Natalie Hobson, Kenny Jacobs, Kurt Kuehn, Neil Lyall, Kira McCabe, Ben Minor, John Nardo, Ted Shifrin, Jim Stankewicz, Glenn Stevens, Gracie Thompson, Lola Thompson, Philip Tiu, Lisa Townsley, John Voight, and Nathan Walters.

To my advisers, Jonathan Hanke and Daniel Krashen, I owe more than I can express. They have shown by example that it is possible to be both a kind, generous human being as well as a diligent, serious mathematician. I am grateful for all the hours they have given me; this is especially true of Jon Hanke, who for years continued to advise me through less-than-ideal conditions. I am

honored to be their student.

I would like to end by thanking my parents, Kathy Williams and Ken Thompson. My mother taught me to love learning, and was–by far–the best and most important teacher I have ever had. My father, the most out-spoken feminist I have ever met, taught me the necessity of having a passion for your profession, a general sense of tenacity, and a respect for and satisfaction in a hard day's work.

Contents

List of Figures

List of Tables

Chapter 1

Background

1.1 Quadratic Forms

General references for this section include [6], [28], [32].

Definition 1.1.1. *Let R be a commutative ring (with 1_R). A **quadratic form** Q **over** R is a polynomial $Q(\vec{x}) \in R[x_1, ...x_n]$ of the form*

$$Q(\vec{x}) \quad = \quad \sum_{i \leq j=1}^{n} c_{ij} x_i x_j$$

with $c_{ij} \in R$.

Given two quadratic forms $Q_1 : R^n \to R$, $Q_2 : R^m \to R$ we define the orthogonal sum $Q_3 := Q_1 \perp Q_2$, $Q_3 : R^{m+n} \to R$ by

$$Q_3(x_1, ..., x_n, x_{n+1}, ..., x_{n+m}) = Q_1(x_1, ..., x_n) + Q_2(x_{n+1}, ..., x_{n+m}).$$

It is clear by definition that for $\vec{x} = \vec{0}$, $Q(\vec{x}) = 0$; however, depending upon the form, there may be non-zero vectors which evaluate to 0. We say that Q is **isotropic** if there exists $\vec{x} \neq \vec{0}$ such that $Q(\vec{x}) = 0$. If Q is not isotropic, it is said to be **anisotropic**.

Example 1.1.1. *Let $Q(\vec{x}) = x_1^2 + x_2^2$.*

- *Over $R = \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ or $\mathbb{F}_p$ (for $p \equiv 3 \pmod 4$), Q is anisotropic.*
- *Over $R = \mathbb{C}$ or $\mathbb{F}_p$ (for $p \equiv 1 \pmod 4$), Q is isotropic.*

When $2 \in R$ is not a zero divisor, we can represent Q by a symmetric matrix M_Q:

$$M_Q = [a_{ij}]_{1 \leq i,j \leq n}$$

where $a_{ij} + a_{ji} = 2c_{ij}$ for $i = j$ and where $a_{ij} + a_{ji} = c_{ij}$ otherwise. Note that in particular this means all diagonal entries of M_Q are elements of R (as $\forall i$, $a_{ii} = c_{ii}$); it is only the off-diagonal entries that are allowed denominators. While $M_Q \in \mathbb{M}_n\left(\frac{1}{2}R\right)$, in the case that $M_Q \in \mathbb{M}_n(R)$ we say that Q is **classical**. If $a_{ij} = 0$ for all $i \neq j$ we say that Q is **diagonal**, and we then write $Q = \langle c_{11}, ..., c_{nn} \rangle$.

Example 1.1.2. *Let $R = \mathbb{Z}$.*

- $Q(\vec{x}) = \langle 1, 1 \rangle = x_1^2 + x_2^2 \leftrightarrow M_Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ *is classical (equiv. is **classically integral**).*
- $Q(\vec{x}) = x_1^2 + 3x_1x_2 + x_2^2 \leftrightarrow M_Q = \begin{bmatrix} 1 & 3/2 \\ 3/2 & 1 \end{bmatrix}$ *is not classical (equiv. is **not classically integral**).*

Two n-ary forms Q_1, Q_2 over R are said to be **R-equivalent** if there exists $A \in GL_n(R)$ such that $A^t M_{Q_1} A = M_{Q_2}$.

Theorem 1.1.1. *Let K be a field with characteristic $char(K) \neq 2$. Let Q be a quadratic form over K. Then Q is K-equivalent to a diagonal form, and can be diagonalized via a series of elementary row and column operations.*

N.B. By definition, this shows that Q is K-equivalent to a diagonal form.

Proof. As stated, this theorem can be found in [28, pg. 7]; however, [6, Section 8.3] treats more general cases.

We proceed by induction on n. When $n = 1$, this is trivial. Assume, then, that the statement holds for all $1 \leq m < n$ and suppose $Q : K^n \to K$. For $Q(\vec{x}) = \sum_{1 \leq i,j \leq n} c_{ij} x_i x_j$, the associated symmetric matrix M_Q is defined by

$$M_Q = [a_{ij}] .$$

There are three disjoint cases:

S1. $a_{1j} = a_{i1} = 0$ for all $2 \leq i, j \leq n$. If this holds, then $Q = \langle a_{11} \rangle \perp Q'$ for $Q' : K^{n-1} \to K$. By induction there exists a series of elementary $(n-1)$ row and column operations $\{A'\}$ such that $A' M_{Q'} A'^t$ is diagonal. Set $A = \begin{bmatrix} 1 & 0 \\ 0 & A' \end{bmatrix}$. Then $A M_Q A^t$ is diagonal, with A a product of elementary row and column operations.

S2. $a_{11} \neq 0$, and for some $2 \leq k \leq n$ $a_{k1} = a_{1k} \neq 0$. For each $2 \leq k \leq n$ with $a_{k1} = a_{1k} \neq 0$ let E_k denote the elementary row matrix sending $r_i \mapsto r_i$ for all $i \neq k$ and $r_k \mapsto r_k - \frac{a_{k1}}{a_{11}} r_1$. Let $E = \prod_k E_k$. Then $E M_Q E^t = [b_{ij}]$ where $b_{1\ell} = b_{\ell 1} = 0$ for all $2 \leq \ell \leq n$. Then apply S1 to $M'_Q = [b_{ij}]$.

S3. $a_{11} = 0$, and for some $2 \leq k \leq n$ $a_{k1} = a_{1k} \neq 0$. Let $2 \leq k \leq n$ be minimal with $a_{k1} = a_{1k} \neq 0$. Let E_k denote the elementary row matrix sending $r_i \mapsto r_i$ for all $i > 1$ and $r_1 \mapsto r_1 + r_k$. Then $E_k M_Q E_k^t = [b_{ij}]$ where $b_{11} = a_{1k} + a_{k1} = 2a_{1k} \neq 0$ (by choice of k and characteristic of the field). Then apply S2 to $M'_Q = [b_{ij}]$.

□

We assume that our quadratic forms Q are **non-degenerate**, meaning $\det(M_Q) \neq 0$. We refer to $\det M_Q$ as the **discriminant** of Q. If Q is a quadratic form over a field K (of characteristic not two) then the discriminant is well-defined up to a non-zero square. Note also that any degenerate form is automatically isotropic; however, considering the form $Q(\vec{x}) = x_1^2 - x_2^2$ isotropy does not imply degeneracy.

Let K be a number field (including $K = \mathbb{Q}$). We say that quadratic forms Q_1, Q_2 over $\mathcal{O}_K$ are in the same genus if they are equivalent over $\mathcal{O}_{K_\nu}$ for all places ν (including $\nu | \infty$). Note that it is possible for $\mathcal{O}_K$-inequivalent forms to be in the same genus:

Example 1.1.3. *(taken from [6, pg. 129]) Consider the quadratic forms* $Q_1(\vec{x}) = x_1^2 + 82x_2^2$ *and* $Q_2(\vec{x}) = 2x_1^2 + 41x_2^2$ *over* $\mathcal{O}_K = \mathbb{Z}$*. These forms are not* $\mathbb{Z}$*-equivalent but are in the same genus.*

It is known that quadratic forms in the same genus have the same discriminant, and therefore there are only finitely many equivalence classes in a genus. See [6, Chapter 9] for additional references.

We say that $m \in R$ is **represented** by Q if there exists $\vec{x} \in R^n$ such that $Q(\vec{x}) = m$. If Q represents m for all $m \in R$, we say that Q is **universal**. In the particular case that R is the ring of integers of a totally real number field K ($K = \mathbb{Q}$ allowed), we say that Q is **totally positive (resp. negative) definite** if for all $\vec{x} \neq \vec{0}$, $\sigma_i(Q(\vec{x})) > 0$ (resp. $\sigma_i(Q(\vec{x})) < 0$) for all $\sigma_i : K \hookrightarrow \mathbb{R}$ embeddings. We say that Q is **positive-universal (resp. negative-universal)** if for all $m \in \mathcal{O}_K^+$ (resp. $m \in \mathcal{O}_K^-$) the equation $Q(\vec{x}) = m$ has a solution for $\vec{x}$ an $\mathcal{O}_K$ vector.

Example 1.1.4. *Let* $Q(\vec{x}) = x_1^2 + x_2^2$*.*

- *Over* $R = \mathbb{F}_p, \mathbb{C}$*,* Q *is universal.*
- *Over* $R = \mathbb{Z}$*,* Q *is positive-definite, but is not positive-universal.*

Regarding positive universality of integral quaternary forms, the following results are fundamental.

Theorem 1.1.2. *(a) (Ramanujan, Dickson) There are, up to equivalence, 54 diagonal positive universal quaternary quadratic forms over* $\mathbb{Z}$*.*

(b) (Conway-Schneeberger, Bhargava) Let Q *be a classically integral quadratic form over* $\mathbb{Z}$*. Then* Q *is positive universal if and only if* Q *represents the numbers*

$$1, 2, 3, 5, 6, 7, 10, 14, \text{ and } 15.$$

Up to equivalence, there are 204 *such forms.*

(c) (Bhargava-Hanke) Let Q *be a quadratic form over* $\mathbb{Z}$*. Then* Q *is positive universal if and only if it represents the numbers*

$1, 2, 3, 5, 6, 7, 10, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31, 34, 35, 37, 42, 58, 93, 110, 145, 203,$ *and* 290.

Up to equivalence there are 6436 *such forms.*

1.2 Characters

General sources for this section include [2, Chapter 6], [4, Chapter 1], and [30, Chapter 3].

Definition 1.2.1. *Let* $N \in \mathbb{N}$. *A* ***Dirichlet character modulo*** N *is a group homomorphism* $\chi : (\mathbb{Z}/N\mathbb{Z})^\times \to \mathbb{C}^\times$.

For any two Dirichlet characters χ, ψ modulo N we define

$$(\chi\psi)(n) := \chi(n)\psi(n), \forall n \in (\mathbb{Z}/N\mathbb{Z})^\times.$$

Thus the set of Dirichlet characters modulo N is a multiplicative group, called the **dual group** of $(\mathbb{Z}/N\mathbb{Z})^\times$. The identity of the dual group is the **trivial character**

$$\begin{aligned} \mathbb{1}_N : (\mathbb{Z}/N\mathbb{Z})^\times &\to \mathbb{C}^\times \\ \mathbb{1}_N(n) &= 1. \end{aligned}$$

Moreover, for any χ, define $\overline{\chi}(a) = \overline{\chi(a)}$ for all $a \in (\mathbb{Z}/N\mathbb{Z})^\times$. We make the following observations:

(1) The values taken by any Dirichlet character are complex roots of unity. We say that χ is **quadratic** when $\chi(n) = \pm 1$ for all n.

(2) Every Dirichlet character mod N can be extended to a map $\chi : \mathbb{Z}/N\mathbb{Z} \to \mathbb{C}$ by setting $\chi(n) = 0$ for all n such that $(n, N) > 1$. This then allows a character to be defined as $\chi : \mathbb{N} \cup \{0\} \to \mathbb{C}$ by composing with the natural reduction map. We will follow the standard abuse of terminology and refer to this extended Dirichlet character as a Dirichlet character.

Example 1.2.1. *The Dirichlet characters modulo* 5 *are given by:*

	0	1	2	3	4
$\mathbb{1}$	0	1	1	1	1
χ_2	0	1	-1	-1	1
χ_3	0	1	i	$-i$	-1
χ_4	0	1	$-i$	i	-1

For $0 < d|N$, we can lift a Dirichlet character modulo d to a Dirichlet character modulo N by setting

$$\chi_N := \chi_d \circ \pi_{N,d}$$

where $\pi_{N,d} : (\mathbb{Z}/N\mathbb{Z})^\times \to (\mathbb{Z}/d\mathbb{Z})^\times$ is the natural projection map; however, while there is always a way to lift characters, there may not always be a way to reduce characters. The **conductor** of χ is defined to be the smallest positive divisor d of N such that $\chi = \chi_d \circ \pi_{N,d}$. When $d = N$, we say χ is a **primitive** character.

Example 1.2.2. *(For details see [4, Sections 1.1, 1.7]) Suppose K is a quadratic number field with discriminant D. There exists a quadratic Dirichlet character χ_D of conductor $|D|$ such that for a prime $p \in \mathbb{Z}$:*

$$\chi_D(p) \quad = \quad \begin{cases} 1, & \text{if } p \text{ splits,} \\ 0, & \text{if } p \text{ ramifies,} \\ -1, & \text{if } p \text{ is inert.} \end{cases}$$

To a given Dirichlet character modulo N, we can define a corresponding **Gauss sum** as

$$\tau(\chi, n) \quad := \quad \sum_{a=0}^{N} \chi(a)\zeta_N^{na}$$

where $\zeta_N = e^{2\pi i/N}$ is a primitive N^{th} root of unity.

We now refer to two well-known results, both of which are proved in [2, Chapter 8]. The first is an orthogonality result regarding $\tau(\mathbb{1}, 1)$:

Theorem 1.2.1.

$$\sum_{a=0}^{N} \zeta_N^{a(x-y)} = \begin{cases} 0, & \text{if } x \not\equiv y \pmod{N} \\ N, & \text{if } x \equiv y \pmod{N} \end{cases}$$

Using that, one can show the following:

Theorem 1.2.2. *Let χ be a Dirichlet character of conductor N. Then $|\tau(\chi, 1)| = \sqrt{N}$.*

For the generalization of characters to number fields, we borrow heavily from [30, Chapter 3] and especially [4, Section 1.7].

Let K be a totally real number field with n real embeddings. Let $\mathfrak{N}$ be a nonzero ideal of $\mathcal{O}_K$. Let $\chi_{\mathfrak{N}} : (\mathcal{O}_K/\mathfrak{N})^\times \to \mathbb{C}^\times$ be a character. As before, we say the character $\chi_{\mathfrak{N}}$ is **primitive** if it does not come from a composition with a natural projection map; that is, if it is not of the form $\chi_{\mathfrak{d}} \circ \pi_{\mathfrak{N},\mathfrak{d}}$ for $\mathfrak{d}|\mathfrak{N}$ a nonzero ideal. One can extend $\chi_{\mathfrak{N}}$ to $\mathcal{O}_K$ by

$$\chi_{\mathfrak{N}}(\mathfrak{a}) = \begin{cases} \chi_{\mathfrak{N}}(\mathfrak{a} \pmod{\mathfrak{N}}), & (\mathfrak{a}, \mathfrak{N}) = 1 \\ 0, & \text{otherwise.} \end{cases}$$

To extend this to a character of principal ideals, we need to ensure that $\chi_{\mathfrak{N}}$ is trivial on units. To this end let $t_1, ..., t_n$ be purely imaginary complex parameters with $\sum_{j=1}^{n} t_j = 0$. Let $u_1, ..., u_n \in \mathbb{R}$ with $u_j \in \{0, 1\}$. We define characters χ_j of $\mathbb{R}^\times$ by

$$\chi_j(a) = \operatorname{sgn}(x)^{u_j} |x|^{t_j}$$

and a character χ_∞ on $\mathcal{O}_K^\times$ by

$$\chi_\infty(a) = \prod_{j=1}^{n} \chi_j(a^{(j)}).$$

One can choose the $\{u_j, t_j\}$ such that $\chi(a) = \chi_\infty(a)\chi_{\mathfrak{N}}(a)$ is trivial on $\mathcal{O}_K^\times$ (see [4, pg. 77] for specifics). We say that a **Hecke character** is a character of ideals whose restriction to principal

ideals arises in this way.

N.B. When $K = \mathbb{Q}$ this does indeed restrict to the case of Dirichlet characters discussed above.

1.3 L-functions

A general reference for this section (from which we will borrow liberally) is [2, Chapter 11-12].

Let $s \in \mathbb{C}$. If χ is a Dirichlet character, we can define an L-function (over $\mathbb{Q}$) as

$$L_{\mathbb{Q}}(s, \chi) \quad := \quad \sum_{n=1}^{\infty} \chi(n) n^{-s}.$$

Using the notation of [2], we set

$$s = \sigma + it$$

for $\sigma, t \in \mathbb{R}$. If $\sigma \geq a$ then

$$\left| \frac{\chi(n)}{n^s} \right| \leq \frac{|\chi(n)|}{n^a}.$$

Therefore, if $\sum \chi(n) n^{-s}$ converges absolutely for $s = a + bi$, then it also converges absolutely for all s with $\sigma \geq a$. Because $|\chi|$ is bounded for any choice of character χ, $\sum \chi(n) n^{-s}$ converges absolutely for $\sigma > 1$ [2, pg. 225]. Moreover, since χ is completely multiplicative for $\sigma > 1$ we have

$$\begin{aligned} L_{\mathbb{Q}}(s, \chi) \quad &:= \quad \sum_{n=1}^{\infty} \chi(n) n^{-s} \\ &= \quad \prod_{p \text{ prime}} \left(1 - \frac{\chi(p)}{p^s} \right)^{-1} \end{aligned}$$

where p runs over all positive prime numbers. (Again, see [2, Theorem 11.7].)

Example 1.3.1. $L_{\mathbb{Q}}(2, \mathbb{1}) = \zeta_{\mathbb{Q}}(2) = \dfrac{\pi^2}{6}$.

L-functions over number fields K can also be defined in a similar manner. Given a Hecke character χ,

$$L_K(s,\chi) = \prod_{\mathfrak{p} \text{ prime}} \left(1 - \frac{\chi(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1}$$

where the product runs over all non-zero prime ideals $\mathfrak{p}$ of $\mathcal{O}_K$.

In certain cases, there are ways to write (and consequently, evaluate) L-functions over number fields as a product of finitely many L-functions over $\mathbb{Q}$.

Example 1.3.2. *Let K be a quadratic number field with discriminant D. Then*

$$\zeta_K(s) = \zeta_{\mathbb{Q}}(s) L_{\mathbb{Q}}(s, \chi_D).$$

where χ_D is the quadratic Dirichlet character of conductor $|D|$ defined in the previous section.

N.B. We will use this example in the case $K = \mathbb{Q}(\sqrt{5})$ later in this document.

Proof.

$$\begin{aligned}
\zeta_K(s) &= \prod_{\mathfrak{p}} \left(1 - \frac{1}{N(\mathfrak{p})^s}\right)^{-1} \\
&= \left(\prod_{p \text{ inert}} \left(1 - \frac{1}{p^{2s}}\right)^{-1}\right) \left(\prod_{p \text{ split}} \left(1 - \frac{1}{p^s}\right)^{-2}\right) \left(\prod_{p \text{ ramifies}} \left(1 - \frac{1}{p}\right)^{-1}\right) \\
&= \left(\prod_{p} \left(1 - \frac{1}{p^s}\right)^{-1}\right) \left(\prod_{p \text{ inert}} \left(1 + \frac{1}{p^s}\right)^{-1}\right) \left(\prod_{p \text{ split}} \left(1 - \frac{1}{p^s}\right)^{-1}\right) \\
&= \zeta_{\mathbb{Q}}(s) \left(\prod_{\chi_D(p)=-1} \left(1 - \frac{\chi_D(p)}{p^s}\right)^{-1}\right) \left(\prod_{\chi_D(p)=1} \left(1 - \frac{\chi_D(p)}{p^s}\right)^{-1}\right) \\
&= \zeta_{\mathbb{Q}}(s) L_{\mathbb{Q}}(s, \chi_D).
\end{aligned}$$

□

As a slight generalization: let $N(\cdot)$ denote the norm map from $\mathcal{O}_K$ to $\mathbb{Z}$, and let ψ be a Dirichlet character on $\mathbb{Q}$. Then

$$L(s, \psi \circ N) = L(s,\psi) L(s, \psi\chi_D).$$

Proof. Let $\mathfrak{p} \in \mathcal{O}_K$ be a prime with $p \in \mathbb{Z}$ a positive generator of the ideal lying below $\mathfrak{p}$. Then

$$\begin{aligned} L_K(s, \psi \circ N) &= \prod_{\mathfrak{p}} \left(1 - \frac{(\psi \circ N)(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1} \\ &= \prod_{\mathfrak{p}|p \text{ ramified}} \left(1 - \frac{(\psi \circ N)(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1} \prod_{\mathfrak{p}|p \text{ inert}} \left(1 - \frac{(\psi \circ N)(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1} \prod_{\mathfrak{p}|p \text{ split}} \left(1 - \frac{(\psi \circ N)(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1}. \end{aligned}$$

We treat each of these three components separately:

For a prime $\mathfrak{p}$ lying above a ramified prime p, $N(\mathfrak{p}) = p$. Moreover

$$\begin{aligned} \prod_{\mathfrak{p}|p \text{ ramified}} \left(1 - \frac{(\psi \circ N)(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1} &= \prod_{p \text{ ramified}} \left(1 - \frac{\psi(p)}{p^s}\right)^{-1} \\ &= \prod_{p \text{ ramified}} \left(1 - \frac{\psi(p)}{p^s}\right)^{-1} \left(1 - \frac{\psi\chi_D(p)}{p^s}\right)^{-1}. \end{aligned}$$

For $p = \mathfrak{p}$ inert, $N(\mathfrak{p}) = p^2$ and

$$\begin{aligned} \prod_{\mathfrak{p}|p \text{ inert}} \left(1 - \frac{(\psi \circ N)(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1} &= \left(\prod_{p \text{ inert}} 1 - \frac{\psi(p^2)}{p^{2s}}\right)^{-1} \\ &= \left(\prod_{p \text{ inert}} 1 - \frac{\psi(p)^2}{p^{2s}}\right)^{-1} \\ &= \left(\prod_{p \text{ inert}} 1 - \frac{\psi(p)}{p^s}\right)^{-1} \left(\prod_{p \text{ inert}} 1 - \frac{\psi\chi_D(p)}{p^s}\right)^{-1}. \end{aligned}$$

Last, we consider split primes. For $\mathfrak{p}|p$ which is split, $N(\mathfrak{p}) = p$. Thus,

$$\begin{aligned} \prod_{\mathfrak{p}|p \text{ split}} \left(1 - \frac{(\psi \circ N)(\mathfrak{p})}{N(\mathfrak{p})^s}\right)^{-1} &= \left(\prod_{p \text{ split}} \left(1 - \frac{\psi(p)}{p^s}\right)^{-1}\right)^2 \\ &= \left(\prod_{p \text{ split}} \left(1 - \frac{\psi(p)}{p^s}\right)^{-1}\right) \left(\prod_{p \text{ split}} \left(1 - \frac{\psi\chi_D(p)}{p^s}\right)^{-1}\right). \end{aligned}$$

This then shows that indeed $L_K(s, \psi \circ N) = L_{\mathbb{Q}}(s, \psi) L_{\mathbb{Q}}(s, \psi\chi_D)$. □

1.4 Quaternion Algebras

General references include [41], [33], [42]. One can also see [34, Chapters 3-4] and [28, Chapter 3].

Definition 1.4.1. *A **division ring** (or **skew field**) is a ring where every non-zero element has a multiplicative inverse.*

Note: A division ring differs from a field in that multiplication in a division ring need not be commutative. Thus, all fields are division rings, but not all division rings are fields.

Definition 1.4.2. *Let F be a field with characteristic $char(F) \neq 2$. Let $u, v \in F^{\times}$. A **quaternion algebra** $B = \left(\frac{u,v}{F}\right)$ over F is a central simple algebra with K-basis $\{1, i, j, ij\}$ satisfying $i^2 = u$, $j^2 = v$, $ij = -ji$.*

Remark: In the case that $char(F) = 2$, one can define a quaternion algebra $B = \left(\frac{u,v}{F}\right)$ by requiring that $u \in F$, $v \in F^{\times}$ and that the non-trivial generators i, j satisfy $i^2 = u$, $j^2 = v$, $ji = (i+1)j$.

Theorem 1.4.1. *(Artin-Wedderburn) Suppose R is a semisimple ring. Then there exist division algebras $D_1, ..., D_t$ and natural numbers $n_1, ..., n_t$ with*

$$R \cong M_{n_1}(D_1) \times ... \times M_{n_t}(D_t).$$

Proof. See [24, pg. 203]. □

Corollary 1.4.1. *As a quaternion algebra B over a field F is a central simple algebra of degree 4, either:*

- *$B \cong D$ for D a four-dimensional division ring over F.*
- *$B \cong M_2(F)$.*

Example 1.4.1. *Let F be any field. $M_2(F)$ is always a quaternion algebra over F; in particular, $M_2(F) \cong \left(\frac{1,1}{F}\right)$ with*

$$\begin{aligned} j &\mapsto \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ i &\mapsto \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, & char(F) \neq 2 \\ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, & char(F) = 2. \end{cases} \end{aligned}$$

Example 1.4.2. *Up to isomorphism, the two quaternion algebras over $F = \mathbb{R}$ are*

1. *The traditional Hamiltonians $\mathbb{H} = \left(\frac{-1,-1}{\mathbb{R}}\right)$.*
2. *The matrix group $M_2(\mathbb{R}) \cong \left(\frac{1,1}{\mathbb{R}}\right)$.*

For the remainder of this section, let F be a number field (including $F = \mathbb{Q}$). Let B be a quaternion algebra over F. Additionally, we refer to a given $b \in B$ as

$$b = \alpha + \beta i + \gamma j + \delta ij$$

where $\alpha, \beta, \gamma, \delta \in F$.

Definition 1.4.3. *There is a standard involution on B (i.e., an anti-automorphism of order 2, see [36, Section 19]) given by*

$$b = \alpha + \beta i + \gamma j + \delta ij \mapsto \overline{b} = \alpha - \beta i - \gamma j - \delta ij.$$

This gives us two functions from B to F:

- *The **trace** function*

$$Tr_B(x) : x \mapsto x + \overline{x}.$$

- *The **(reduced) norm***

$$N_B(x) : x \mapsto x\overline{x}.$$

Let B_0 denote the trace 0 elements of B; i.e., $B_0 = \{\beta i + \gamma j + \delta ij : \beta, \gamma, \delta \in F\}$.
The norm function is of particular interest as it is a quaternary quadratic form:

$$\begin{aligned} x\overline{x} &= (x_1 + x_2 i + x_3 j + x_4 ij)(x_1 - x_2 i - x_3 j - x_4 ij) \\ &= x_1^2 - ux_2^2 - vx_3^2 + uvx_4^2. \end{aligned}$$

For each completion F_v of F, we record the isotropy of the norm form by the **Hilbert symbol** $(u, v)_{F_v}$. Specifically, we define the Hilbert symbol to be

$$(u, v)_{F_v} := \begin{cases} 1, & N_B(x) = x_1^2 - ux_2^2 - vx_3^2 + uvx_4^2 \text{ is isotropic over } F_v \\ -1, & \text{otherwise.} \end{cases}$$

Theorem 1.4.2. *Let F be a number field with ν denoting a place of F. Let $B = \left(\frac{u,v}{F}\right)$ be a quaternion algebra. The following are equivalent:*

1. $B_\nu = \left(\dfrac{u, v}{F_\nu}\right) \cong M_2(F_\nu)$.
2. $N_B(x) = x_1^2 - ux_2^2 - vx_3^2 + uvx_4^2$ *is isotropic over* F_ν.
3. $N_{B_0}(x) = -ux_2^2 - vx_3^2 + uvx_4^2$ *is isotropic over* F_v.

*When these conditions are satisfied, we say that B is **unramified** (or is **split**) at v. If these conditions are not satisfied, we say that B is **ramified** at v.*

Proof. (3) $\Rightarrow$ (2) is trivial.
(1) $\Rightarrow$ (2) is trivial.
(1) $\Leftrightarrow$ (2) $\Leftrightarrow$ (3) can be found in the classic literature, including [28, Theorem 2.7, pg. 58]. $\square$

Theorem 1.4.3. *(Hilbert) Let F be a number field. Let $u, v \in F$ with $uv \neq 0_F$. Then $(u,v)_{F_\nu} = 1$ for almost all ν and*

$$\prod_{\nu} (u,v)_{F_\nu} = 1.$$

Proof. See [34, Theorem 3, pg. 23] for a detailed proof in the case $F = \mathbb{Q}$ (and for a statement on how to extend to number fields). □

Corollary 1.4.2. *$B = \left(\frac{u,v}{F}\right)$ ramifies at an even number of places of F.*

Theorem 1.4.4. *A quaternion algebra B over a number field F is uniquely determined by the places of F at which it ramifies.*

Proof. The proof relies on specific Hilbert symbol facts and local-global results of ternary quadratic forms. See [34, Chapter 3] for details on the treatment for rational quaternion algebras. See [25, Section 2] for an extremely general treatment. □

Definition 1.4.4. *We define the **discriminant** $\mathfrak{D} = \mathfrak{D}_B$ of a quaternion algebra B over a number field F to be the integral ideal given by the product of the finite primes $\mathfrak{p}$ over which B ramifies.*

Example 1.4.3. *Let F be a number field, and consider the Hamiltonians $B = \left(\frac{-1,-1}{F}\right)$.*

- *If $F = \mathbb{Q}$, then B has discriminant 2.*
- *If $F = \mathbb{Q}(\sqrt{5})$, then B has discriminant 1.*

Definition 1.4.5. *Let B be a quaternion algebra over a number field F. A **quaternion order (over $\mathcal{O}_F$)** is a subring $R \subset B$ that is finitely generated as an $\mathcal{O}_F$-module and such that $RF = B$.*

Quaternion orders always exist: Let B be any quaternion algebra over F with standard generators $\{1, i, j, ij\}$. Then $R = \mathcal{O}_F + \mathcal{O}_F i + \mathcal{O}_F j + \mathcal{O}_F ij$ is a quaternion order.

Definition 1.4.6. *A quaternion order $R \subset B$ is said to be **maximal** if there is no other quaternion order $S \subseteq B$ satisfying $R \subseteq S \subseteq B$.*

One way to verify that a given quaternion order $R \subseteq B$ is maximal is to compare its discriminant to the discriminant $\mathfrak{D}$ of B. The **discriminant** $\mathfrak{d}_R$ of an order R is an ideal satisfying

$$\mathfrak{d}_R^2 = I$$

where I is the ideal generated by the set

$$\{\det(Tr_B(x_i x_j))_{i,j=1,...,4} : x_1, ..., x_4 \in R\}.$$

Theorem 1.4.5. *An order $R \subset B$ is maximal iff $\mathfrak{d}_R = \mathfrak{D}$.*

Proof. A proof is sketched in [42, pg. 29], which in turn references [41, Chapter 2]. □

Example 1.4.4. *Let $B = \left(\frac{-1,-1}{\mathbb{Q}(\sqrt{5})}\right)$. Then $R = \mathcal{O}_F e_1 + \mathcal{O}_F e_2 + \mathcal{O}_F e_3 + \mathcal{O}_F e_4$ is a maximal order where*

$$\begin{aligned} e_1 &= \tfrac{1}{2}\left(1 - \tfrac{1-\sqrt{5}}{2}i + \tfrac{1+\sqrt{5}}{2}j\right) \\ e_2 &= \tfrac{1}{2}\left(-\tfrac{1-\sqrt{5}}{2}i + j + \tfrac{1+\sqrt{5}}{2}k\right) \\ e_3 &= \tfrac{1}{2}\left(\tfrac{1+\sqrt{5}}{2}i - \tfrac{1-\sqrt{5}}{2}j + k\right) \\ e_4 &= \tfrac{1}{2}\left(i + \tfrac{1+\sqrt{5}}{2}j - \tfrac{1-\sqrt{5}}{2}k\right). \end{aligned}$$

1.5 (Classical) Modular Forms

General background and details to proofs can be found in [14], [30], [35] and [39]. Any specific additional references will be given in context. We intend to introduce terminology in such a way as to highlight parallels with the next section.

We refer to $SL_2(\mathbb{Z})$ (or $GL_2^+(\mathbb{Z})$) as the **full modular group** $\Gamma(1)$. Let $0 \neq N \subset \mathbb{Z}$ be an ideal (as a slight, yet common, abuse of notation in this section we will also let N refer to the positive element which generates the ideal N). We define the **principal congruence subgroup of level**

N to be

$$\Gamma(N) := \left\{ \gamma \in SL_2(\mathbb{Z}) : \gamma \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \pmod{N} \right\}.$$

Any subgroup Γ of $GL_2^+(\mathbb{Q})$ such that Γ contains some $\Gamma(N)$ with finite index is a **congruence subgroup** (of $GL_2^+(\mathbb{Q})$).

Example 1.5.1. *Two additional congruence subgroups (of level N) which will be of importance are*

- $\Gamma_1(N) = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in SL_2(\mathbb{Z}) : \begin{bmatrix} a & b \\ c & d \end{bmatrix} \equiv \begin{bmatrix} 1 & * \\ 0 & 1 \end{bmatrix} \pmod{N} \right\}$
- $\Gamma_0(N) = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in SL_2(\mathbb{Z}) : \begin{bmatrix} a & b \\ c & d \end{bmatrix} \equiv \begin{bmatrix} * & * \\ 0 & * \end{bmatrix} \pmod{N} \right\}.$

  Note also that we have

$$\Gamma(N) \subset \Gamma_1(N) \subseteq \Gamma_0(N) \subseteq SL_2(\mathbb{Z}).$$

We next discuss the relative indices of these subgroups. Noting that the natural map $SL_2(\mathbb{Z}) \twoheadrightarrow SL_2(\mathbb{Z}/N\mathbb{Z})$ has kernel $\Gamma(N)$, by the Orbit-Stabilizer Theorem

$$[SL_2(\mathbb{Z}) : \Gamma(N)] = |SL_2(\mathbb{Z}/N\mathbb{Z})| = N^3 \prod_{p|N} \left(1 - \frac{1}{p^2}\right).$$

Similarly,

$$\begin{array}{rcl} \Gamma_1(N) & \twoheadrightarrow & \mathbb{Z}/N\mathbb{Z} \\ \begin{bmatrix} a & b \\ c & d \end{bmatrix} & \mapsto & b \pmod{N} \end{array}$$

has kernel $\Gamma(N)$, implying $[\Gamma_1(N) : \Gamma(N)] = N$. And

$$\begin{array}{rcl} \Gamma_0(N) & \twoheadrightarrow & (\mathbb{Z}/N\mathbb{Z})^* \\ \begin{bmatrix} a & b \\ c & d \end{bmatrix} & \mapsto & d \pmod{N} \end{array}$$

has kernel $\Gamma_1(N)$, so $[\Gamma_0(N) : \Gamma_1(N)] = \phi(N)$.

In conclusion, we have:

$$\underbrace{SL_2(\mathbb{Z}) \overbrace{\supset}^{{}_N\prod_{p|N}(1+\frac{1}{p})} \Gamma_0(N) \underbrace{\supseteq}_{{}_N\prod_{p|N}(1-\frac{1}{p})=\phi(N)} \Gamma_1(N) \overbrace{\supseteq}^{N} \Gamma(N)}_{{}_{N^3}\prod_{p|N}(1-\frac{1}{p^2})}$$

The group $GL_2^+(\mathbb{Q})$ acts on $\mathbb{Q} \cup \{\infty\}$ by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \left(\frac{m}{n}\right) = \frac{am+bn}{cm+dn}.$$

Moreover, $SL_2(\mathbb{Z})$ is a subgroup of $GL_2^+(\mathbb{Q})$ which acts transitively on $\mathbb{Q} \cup \{\infty\}$ (as any rational number can take the form $\frac{a}{c}$ where $(a, c) = 1$, which then by definition of relatively prime means there exist $b, d \in \mathbb{Z}$ with $ad - bc = 1$).

For Γ a congruence subgroup, we say the orbits of $\Gamma\backslash(\mathbb{Q} \cup \{\infty\})$ are the **cusps** of Γ. While we have just shown that for $\Gamma = SL_2(\mathbb{Z})$ there is one cusp, we in fact have:

Theorem 1.5.1. *For any congruence subgroup $\Gamma \subseteq SL_2(\mathbb{Z})$, Γ has finitely many cusps.*

Proof. See [14, pg. 58]. Note that this really is an application of the Orbit-Stabilizer Theorem and the transitivity of $SL_2(\mathbb{Z})$ on $\mathbb{Q} \cup \{\infty\}$. □

Definition 1.5.1. *A function $f : \mathbb{H} \to \mathbb{C}$ is called a **modular form for** $\Gamma \subseteq SL_2(\mathbb{Z})$, $[SL_2(\mathbb{Z}) : \Gamma] < \infty$ **of weight** $k \in \mathbb{Z}^{\geq 0}$ **and level** N if it satisfies all of the following conditions:*

I. f is analytic on $\mathbb{H}$.

II. $f(\gamma z) = f\left(\dfrac{az+b}{cz+d}\right) = (cz+d)^k f(z)$ *for all* $\gamma = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \Gamma$ *and* $z \in \mathbb{H}$.

III. f *is holomorphic at all cusps.*

We denote the space of such forms by $\mathcal{M}_k(\Gamma)$. *If* $f \in \mathcal{M}_k(\Gamma)$ *satisfies the additional condition*

IV. f *vanishes at all cusps*

then we say f *is a* ***cusp form****. We denote the space of such forms by* $\mathcal{S}_k(\Gamma)$.

Notes:

- Every congruence subgroup Γ of level N contains $\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$ for some $1 \leq a \leq N$. This means that every modular form f is $a\mathbb{Z}$-periodic for some a. Coupled with holomorphy at all cusps, f has a Fourier expansion of the form

$$f(z) = \sum_{n \geq 0} a_n e^{2\pi i n z/a}.$$

  Moreover, if $f \in S_k(\Gamma)$ then vanishing at all cusps forces $a_0 = 0$.

  Example 1.5.2. *For* $f \in \mathcal{M}_k(\Gamma_1(N))$, *because* $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \in \Gamma_1(N)$

$$\begin{aligned} f(z) &= \sum_{n \geq 0} a_n e^{2\pi i n z} \\ &= \sum_{n \geq 0} a_n q^n \end{aligned}$$

  where $q := e^{2\pi i z}$.

- Given $\Gamma = \Gamma_0(N)$ and a Dirichlet character χ of modulus N we can define

$$\begin{aligned} \chi : \Gamma &\to \mathbb{C}^\times \\ \begin{bmatrix} a & b \\ c & d \end{bmatrix} &\mapsto \chi(d). \end{aligned}$$

We then can refer to the spaces $\mathcal{M}_k(\Gamma_0(N), \chi), \mathcal{S}_k(\Gamma_0(N), \chi)$, where properties I, III, and IV above still hold but where now the forms satisfy

II′. $f(\gamma z) = \chi(\gamma)(cz+d)^k f(z)$ for all $\gamma = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \Gamma_0(N)$ and $z \in \mathbb{H}$.

Theorem 1.5.2. *For Γ a congruence subgroup $\mathcal{M}_k(\Gamma)$ is a finite-dimensional $\mathbb{C}$-vector space.*

Proof. See [14, Chapter 3]. For examples of generators for spaces $\mathcal{M}_k(\Gamma(1))$ for various k see [18, pg. 27]. □

The spaces $S_k(\Gamma)$ are equipped with an inner product, called the **Petersson inner product**. This inner product extends to all $f, g \in \mathcal{M}_k(\Gamma)$, so long as at least one of f or g is a cusp form [14, pg. 182-183]. We define the subspace of **Eisenstein series**, denoted $E_k(\Gamma)$, as the orthogonal complement of $S_k(\Gamma)$ with respect to this inner product.

Theorem 1.5.3. *Fixing a weight $k \in \mathbb{Z}^{\geq 0}$, and congruence subgroup Γ*

$$\mathcal{M}_k(\Gamma) = E_k(\Gamma) \oplus S_k(\Gamma),$$

where the subspaces $E_k(\Gamma)$ and $S_k(\Gamma)$ are orthogonal with respect to the Petersson inner product.

Proof. See [30, Theorem 2.1.7]. □

For the remainder of this section, we suppose Γ is either $\Gamma_1(N)$ or $\Gamma_0(N)$ for some $N \in \mathbb{N}$.

Theorem 1.5.4. *For a positive integer N, allowing χ to run over all Dirichlet characters of modulus N, we have*

$$\mathcal{M}_k(\Gamma_1(N)) \quad = \quad \oplus_\chi \mathcal{M}_k(\Gamma_0(N), \chi).$$

Proof. See [30, Lemma 4.3.1]. □

The spaces $\mathcal{M}_k(\Gamma_0(N), \chi)$ are equipped with the action of **Hecke operators**. For p a nonzero prime (generated by the element p), we define the action of the Hecke operator T_p on $f \in$

$\mathcal{M}_k(\Gamma_0(N), \chi)$ by the formula:

$$(T_p f)(z) = \sum_{n=0}^{\infty} \left(a_{np}(f) + \chi(p) p^{k-1} a_{n/p}(f) \right) e^{2\pi i n z}$$

where $n/p = 0$ when $n \not\equiv 0 \pmod{p}$.

Theorem 1.5.5. *Let $p, q \nmid N$ be nonzero primes.*

1. *If $r, s \in \mathbb{N}$ then for all $f \in \mathcal{M}_k(\Gamma_0(N), \chi)$:*
   - $(T_{p^r} T_{q^s})(f) = (T_{q^s} T_{p^r})(f)$.
   - $T_{p^r}(f) = (T_p T_{p^{r-1}} - p^{k-1}\chi(p) T_{p^{r-2}})(f)$.
2. *The Hecke operators $\{T_p\}$ act by diagonalizable matrices on the spaces $M_k(\Gamma)$.*

Proof. See [14, Section 5.2-5.4], and [39, §2.4, 3.4]. □

Let $M|N$ and $t|(N/M)$ be natural numbers; let $q = e^{2\pi i z}$. We can realize a cusp form of level M to a cusp form of level N in the following sense:

$$\begin{aligned} \alpha_t : S_k(\Gamma_1(M)) &\to S_k(\Gamma_1(N)) \\ f(q) &\mapsto f(q^t) \end{aligned}$$

We define the **old subspace of** $\Gamma_1(N)$, denoted $\Gamma_1(N)_{\text{Old}}$ to be

$$\sum_{t|(N/M)} \sum_{M|N} \alpha_t(S_k(\Gamma_1(M))).$$

The **new subspace of** $\Gamma_1(N)$, denoted $S_k(\Gamma_1(N))_{\text{New}}$, is the orthogonal complement of the old space with respect to the Petersson inner product. That is,

$$S_k(\Gamma_1(N)) = S_k(\Gamma_1(N))_{\text{New}} \oplus S_k(\Gamma_1(N))_{\text{Old}}.$$

Theorem 1.5.6. *The old and new subspaces are preserved by the Hecke operators.*

Proof. See [14, Proposition 5.6.2]. □

We define a **newform** $f(z) = \sum_{n=1}^{\infty} a_n q^n \in S_k(\Gamma_1(N))_{\text{New}}$ to be a Hecke eigenform normalized so that $a_1 = 1$.

1.6 Hilbert Modular Forms

General background and details to proofs can be found in [15], [16], [17], and [37]. Any specific additional references will be given in context. Unless otherwise specified, we assume K is a totally real number field with distinct embeddings $\sigma_1, ..., \sigma_n$. We also assume that the **narrow class number** is 1. (Recall that the narrow class group of K is the group of all fractional ideals modulo principal ideals with totally positive generator. The requirement that K have class number 1 means that every nonzero ideal has a totally positive generator.).

Let $GL_2^+(\mathcal{O}_K)$ denote the 2×2 matrices with entries in $\mathcal{O}_K$ and totally positive determinant. This is called the **full Hilbert modular group** (attached to K). Let $0 \neq \mathfrak{N} \subset \mathcal{O}_K$ be an ideal. $\Gamma(\mathfrak{N}) = \{\gamma \in GL_2^+(\mathcal{O}_K) : \gamma \equiv 1_2 \pmod{\mathfrak{N}}\}$ is the **principal congruence subgroup of level $\mathfrak{N}$.** Let $Z(\mathcal{O}_K)$ be the center of $GL_2^+(\mathcal{O}_K)$. Any $\Gamma \subset GL_2^+(K)$ such that $[Z(\mathcal{O}_K)\Gamma : \Gamma(\mathfrak{N})] < \infty$ is a congruence subgroup.

Example 1.6.1. *For a nonzero ideal $\mathfrak{N} \subset \mathcal{O}_K$ we define*

$$\Gamma_0(\mathfrak{N}) := \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \begin{pmatrix} \mathcal{O}_K & \mathcal{O}_K \\ \mathfrak{N} & \mathcal{O}_K \end{pmatrix} : ad - bc \in \mathcal{O}_K^{\times +} \right\}.$$

This is a congruence subgroup of $GL_2^+(K)$ of level $\mathfrak{N}$.

The cusps of the Hilbert modular group are the elements $\alpha = (\sigma_1(\alpha), ..., \sigma_n(\alpha)) \in K^n$ together with $\infty = (i\infty, ..., i\infty)$. We say that two cusps are inequivalent if their orbits under Γ are disjoint.

Definition 1.6.1. *A function $f : \mathbb{H}^n \to \mathbb{C}$ is called a **Hilbert modular form for** Γ **of weight** $k = (k_1, ..., k_n) \in (2\mathbb{Z})^n$ **and level** $\mathfrak{N}$ if it satisfies all of the following conditions*

I. f *is holomorphic on* $\mathbb{H}^n$.

II. $f\left(\dfrac{\sigma_1(a)z_1+\sigma_1(b)}{\sigma_1(c)z_1+\sigma_1(d)},...,\dfrac{\sigma_n(a)z_n+\sigma_n(b)}{\sigma_n(c)z_n+\sigma_n(d)}\right)=\left(\prod_{i=1}^{n}(\sigma_i(c)z_i+\sigma_i(d))^{k_i}\det(\sigma_i(\gamma))^{-k_i/2}\right)f(z_1,...,z_n)$ *for all* $\gamma\in\Gamma$.

We denote the space of all such forms as $M_k(\Gamma)$.

In comparison to classical modular forms, we note that there is no additional holomorphy condition at the cusps. This is because in the Hilbert case it is automatic, following from Koecher's principle ([16, Section 1.4]).

Still we can define a cusp form as a modular form which satisfies the additional condition that

IV. f vanishes at all cusps.

We denote the space of cusp forms by $S_k(\Gamma)$.

Every congruence subgroup Γ of level $\mathfrak{N}$ contains $M_{\mathfrak{N}} := \left\{\mathfrak{a}\in\mathbb{R}^n : \begin{pmatrix}1 & \mathfrak{a}\\ 0 & 1\end{pmatrix}\in\Gamma\right\}$. By construction, $f\in\mathcal{M}_k(\Gamma)$ is $\mathfrak{a}\mathcal{O}_K$-periodic for some $\mathfrak{a}$. Coupled with holomorphy, f then has a Fourier expansion of the form

$$f(z)=a_0+\sum_{v\in M_{\mathfrak{N}}^{\vee +}}a_v e^{2\pi i\,\mathrm{Tr}(v\cdot z)}$$

where

$$M_{\mathfrak{N}}^{\vee}:=\{k\in K:\mathrm{Tr}_{K/\mathbb{Q}}(km)\in\mathbb{Z}\ \forall m\in M_{\mathfrak{N}}\}$$

and

$$\mathrm{Tr}(v\cdot z):=\sigma_1(v)z_1+...+\sigma_n(v)z_n.$$

Notes:

- If $\Gamma = \Gamma_0(\mathfrak{N})$, then $M_{\mathfrak{N}}^{\vee} = \mathfrak{d}^{-1}$ is the inverse different of K [12, pg.6] and the Fourier expansion of f is of the form

$$f(z) = a_0 + \sum_{v \in (\mathfrak{d}^{-1})^+} a_v e^{2\pi i \operatorname{Tr}(vz).}$$

- If we assume $K = \mathbb{Q}$ the Fourier expansion above reduces to the Fourier expansion given in the previous section.

Theorem 1.6.1. *For Γ a congruence subgroup $\mathcal{M}_k(\Gamma)$ is a finite-dimensional $\mathbb{C}$-vector space.*

Proof. See [15, I. §6] □

Acting on the spaces $\mathcal{M}_2(\mathfrak{N})$ are pairwise commuting, diagonalizable Hecke operators $T_{\mathfrak{n}}$ for $0 \neq \mathfrak{n} \subseteq \mathcal{O}_K$ an ideal relatively prime to $\mathfrak{N}$. Suppose $\mathfrak{p} \nmid \mathfrak{N}$ is a nonzero prime ideal of $\mathcal{O}_K$. Since we assume that K has narrow class number one, there is a totally positive generator p of $\mathfrak{p}$. We then define the action of the Hecke operator $T_{\mathfrak{p}}$ by the formula

$$(T_{\mathfrak{p}} f)(z) = N(\mathfrak{p}) f(pz) + \frac{1}{N(\mathfrak{p})} \sum_{a \in \mathcal{O}_K / \mathfrak{p}\mathcal{O}_K} f\left(\frac{z+a}{p}\right).$$

[12, pg. 6]

Theorem 1.6.2. *1. If $\mathfrak{p}$, $\mathfrak{q} \nmid \mathfrak{N}$ are distinct nonzero prime ideals, and if $r, s \in \mathbb{N}$, then*

- $(T_{\mathfrak{p}^r \mathfrak{q}^s})(f) = (T_{\mathfrak{p}^r} T_{\mathfrak{q}^s})(f)$
- $(T_{\mathfrak{p}^r})(f) = (T_{\mathfrak{p}} T_{\mathfrak{p}^{r-1}} - N(\mathfrak{p}) T_{p^{r-2}})(f)$

Proof. See [16, pg. 61]. □

1.7 Theta Series and Local Densities

General references for this section include [21] and [38]. Additional references for specific proofs will also be provided.

Throughout this section let Q be a **quaternary** positive-definite integral quadratic form over a real quadratic number field K (or $K = \mathbb{Q}$). [Note that all definitions can be altered so that Q has three or more variables and is defined over any totally real number field. Note also that we refer to the symmetric matrix associated to Q as M_Q.] We define the **level** $\mathfrak{N}_Q$ of Q to be the largest $\mathcal{O}_K$ ideal such that for all primes $\mathfrak{p}$, $\mathfrak{N}_\mathfrak{p}(2M_{Q,\mathfrak{p}})^{-1}$ is a matrix of integral ideals with diagonal entries in $2\mathcal{O}_K$. We also define the **determinant** $\mathfrak{D}_Q$ of Q to be $\det(M_Q)$. Last, we set the following Hecke character defined for all $\mathfrak{p} \nmid 2\mathfrak{N}_Q$ by

$$\chi_Q(\mathfrak{p}) = \left(\frac{\mathfrak{D}_Q}{\mathfrak{p}}\right).$$

Recall that the local normalized form of Q is

$$Q(\vec{x}) \equiv_{\mathcal{O}_{K,\mathfrak{p}}} \sum_j \pi_\mathfrak{p}^{v_j} Q_j(\vec{x}_j),$$

with $\dim(Q_j) \leq 2$ (and in fact, for $\mathfrak{p} \nmid 2$, $\dim(Q_j) = 1$). We then define

$$\mathbb{S}_0 = \{j | v_j = 0\} \qquad \mathbb{S}_1 = \{j | v_j = 1\} \qquad \mathbb{S}_2 = \{j | v_j \geq 2\},$$

and we let $s_i = \sum_{j \in S_i} \dim(Q_j)$.

For fixed $m \in \mathcal{O}_K^+$, we let

$$r_Q(m) := \#\{\vec{x} \in \mathcal{O}_K^4 : Q(\vec{x}) = m\}.$$

We then define the **theta series** associated to Q as

$$\Theta_Q(z) = 1 + \sum_{m \in (\mathfrak{d}^{-1})^+} r_Q(m) e^{2\pi i Tr(m \cdot z)}.$$

Theorem 1.7.1. *$\Theta_Q(m)$ is a Hilbert modular form of weight $k = 2$ over $\Gamma_0(\mathfrak{N}_Q)$ with associated character χ_Q.*

Proof. See [1, Theorem 2.2, pg. 61]. □

As each space of modular forms of fixed weight, level and character decomposes into a direct sum of Eisenstein series and cusp forms, we write $\Theta_Q(z) = E_Q(z) \oplus S_Q(z)$, and

$$r_Q(m) = a_E(m) + a_S(m)$$

for each coefficient in the Fourier expansion of Θ_Q.

Following the language of [21] and [38], for fixed $m \in \mathcal{O}_K^+$ and fixed totally positive definite form Q over $\mathcal{O}_K$ we define the **local density** $\beta_\nu(m)$ at a place ν of K by

$$\beta_\nu(m) := \lim_{U \to \{m\}} \frac{\mathrm{Vol}(Q^{-1}(U))}{\mathrm{Vol}(U)}$$

where U is an open neighborhood of m in K_ν and where the volume is determined by a fixed Haar measure.

Siegel's product formula [38, Sections 10, 11] then states

$$a_E(m) = \prod_\nu \beta_\nu(m).$$

When $\nu|\infty$, explicit values for $\beta_\nu(m)$ can be computed again using results of Siegel. Specifically

$$\prod_{\nu|\infty} \beta_\nu(m) = \pi^{2[K:\mathbb{Q}]} \Delta(K)^{-3/2} N_{K/\mathbb{Q}}(\det(M_Q))^{-1/2} N_{K/\mathbb{Q}}(m).$$

Note that this is a special case of Hilfssatz 72 of [38].

When $\nu \nmid \infty$ and we associate the prime ideal $\mathfrak{p}$ with its place $\nu_\mathfrak{p}$, $\beta_\mathfrak{p}(m)$ can be rewritten as

$$\beta_\mathfrak{p} = \lim_{\nu \to \infty} \frac{r_{\mathfrak{p}^\nu}(m)}{N(\mathfrak{p})^{3\nu}}$$

where $N(\mathfrak{p})$ is the norm of the ideal $\mathfrak{p}$ and where $r_{\mathfrak{p}^\nu}(m) := \#\left\{\vec{x} \in (\mathcal{O}_K/\mathfrak{p}^\nu\mathcal{O}_K)^4 \,|Q(\vec{x}) \equiv m \pmod{\mathfrak{p}^\nu}\right\}$ ([21, pg. 368]–note that this reference has a typo in the power of $N(\mathfrak{p})$. What appears here is correct.).

To count $r_{\mathfrak{p}^v}(m)$ for $v >> 1$, we want to use Hensel's Lemma and reduction maps. Following the terminology of [21]:

Definition 1.7.1. *Let $R_{\mathfrak{p}^v}(m) := \{\vec{x} \in (\mathcal{O}_K/\mathfrak{p}^v\mathcal{O}_K)^4 : Q(\vec{x}) \equiv m \pmod{\mathfrak{p}^v}\}$. Note that by definition $\#R_{\mathfrak{p}^v}(m) = r_{\mathfrak{p}^v}(m)$.*

*$\vec{x} \in R_{\mathfrak{p}^v}(m)$ is of **Zero type** if $\vec{x} \equiv \vec{0} \pmod{\mathfrak{p}}$ (in which case, we say $\vec{x} \in R^{\text{Zero}}_{\mathfrak{p}^v}(m)$ with $\#R^{\text{Zero}}_{\mathfrak{p}^v}(m) := r^{\text{Zero}}_{\mathfrak{p}^v}(m)$), is of **Good type** if $\mathfrak{p}^{v_j}\vec{x_j} \not\equiv \vec{0} \pmod{\mathfrak{p}}$ for some j (in which case, we say $\vec{x} \in R^{\text{Good}}_{\mathfrak{p}^v}(m)$ with $\#R^{\text{Good}}_{\mathfrak{p}^v}(m) := r^{\text{Good}}_{\mathfrak{p}^v}(m)$), and is of **Bad type** otherwise.*

As for the reduction maps:

Theorem 1.7.2.

$$r^{\text{Good}}_{\mathfrak{p}^{k+\ell}}(m) = N(\mathfrak{p})^{3\ell} r^{\text{Good}}_{\mathfrak{p}^k}(m)$$

for $k \geq 2\text{ord}_{\mathfrak{p}}(2) + 1$.

Proof. See [21, Lemma 3.2]. □

Theorem 1.7.3. *The map*

$$\begin{aligned} \pi_Z : R^{\text{Zero}}_{\mathfrak{p}^k}(m) &\to R_{\mathfrak{p}^{k-2}}\left(\frac{m}{\pi_{\mathfrak{p}}^2}\right) \\ \vec{x} &\mapsto \pi_{\mathfrak{p}}^{-1}\vec{x} \pmod{\mathfrak{p}^{k-2}} \end{aligned}$$

is a surjective map with multiplicity $N(\mathfrak{p})^4$.

Proof. See [21, pg. 359]. □

Theorem 1.7.4.

- *Bad-Type-I: This occurs when $\mathbb{S}_1 \neq \emptyset$ and $\vec{x}_{\mathbb{S}_1} \not\equiv \vec{0}$.*

$$\pi_{B'} : R^{\text{Bad},\vec{x}_{\mathbb{S}_1}\not\equiv\vec{0}}_{\mathfrak{p}^k,Q}(m) \to R^{\text{Good}}_{\mathfrak{p}^{k-1},Q'}\left(\frac{m}{\pi_{\mathfrak{p}}}\right)$$

which is defined for each index j by

$$\begin{aligned} \vec{x_j} &\mapsto \pi_{\mathfrak{p}}^{-1}\vec{x_j} \quad v'_j = v_j + 1, \quad j \in \mathbb{S}_0 \\ \vec{x_j} &\mapsto \vec{x_j} \qquad v'_j = v_j - 1, \quad j \notin \mathbb{S}_0 \end{aligned}$$

is surjective with multiplicity $N(\mathfrak{p})^{s_1+s_2}$.

- *Bad-Type-II:*

$$\pi_{B''} : R_{\mathfrak{p}^k,Q}^{\mathrm{Bad},\vec{x}_{\mathbb{S}_1}\equiv\vec{0}}(m) \quad \to \quad R_{\mathfrak{p}^{k-2},Q''}^{\vec{x}_{\mathbb{S}_2}\not\equiv\vec{0}}\left(\frac{m}{\pi_{\mathfrak{p}}^2}\right)$$

which is defined for each index j by

$$\begin{aligned} \vec{x_j} &\mapsto \pi_{\mathfrak{p}}^{-1}\vec{x_j} \qquad v''_j = v_j, \qquad j \in \mathbb{S}_0 \cup \mathbb{S}_1 \\ \vec{x_j} &\mapsto \vec{x_j} \qquad v''_j = v_j - 2, \qquad j \notin \mathbb{S}_2 \end{aligned}$$

is surjective with multiplicity $N(\mathfrak{p})^{8-s_0-s_1}$.

Proof. Again, see [21, pg. 360]. □

The last consideration to be made when computing local densities is that of the stable ideal.

Definition 1.7.2. *A number $m \in \mathcal{O}_K^+$ is $\mathfrak{p}$-**stable** if m is locally represented at $\mathfrak{p}$, and for all $k >> 1$ then the quantity*

$$r_{\mathfrak{p}^k}^{\mathrm{Good}}(\pi_{\mathfrak{p}}^{2v})(m) + r_{\mathfrak{p}^k}^{\mathrm{Bad}}(\pi_{\mathfrak{p}}^{2v}m)$$

*is constant for all $v \geq 1$. We say that m is **stable** if it is $\mathfrak{p}$-stable for all primes $\mathfrak{p}$.*

By [21, pg. 362] we know that all m are $\mathfrak{p}$-stable when $\mathfrak{p} \nmid \mathfrak{N}_Q$ and that $\mathfrak{N}_Q\mathcal{O}_K$ is a stable ideal in the sense that m is stable for all locally represented $m \in \mathfrak{N}_Q\mathcal{O}_K$.

1.8 Quaternionic Modular Forms and the Jacquet-Langlands Correspondence

References for this section include [10], [11], [12] and [22]. We borrow heavily from [12]. Throughout, we assume that B is a quaternion algebra over a totally real number field K ($K \neq \mathbb{Q}$, only to highlight Hilbert modular forms) which has narrow class number 1. Additionally, we assume that B splits at all infinite places of K. Last, we let $R \subset B$ denote a maximal order.

Let

$$R_0(\mathfrak{N}) := \left\{ x \in R : x \equiv \begin{pmatrix} * & * \\ 0 & * \end{pmatrix} \pmod{\mathfrak{N}} \right\}.$$

If I and J are two invertible (right) fractional R-ideals, they are in the same **ideal class** if they are isomorphic as right R-modules. The equivalence class of I is denoted by $[I]$ and the set of all invertible right R-modules is $\operatorname{Cl} R$.

Definition 1.8.1. *A **quaternionic modular form** for B of parallel weight 2 and level $\mathfrak{N}$ is a map*

$$f : \operatorname{Cl} R_0(\mathfrak{N}) \to \mathbb{C}.$$

The space of such forms is denoted $\mathcal{M}_2^B(\mathfrak{N})$.

A modular form for B which is orthogonal to the one-dimensional subspace of constant functions [the Eisenstein series equivalent] is called a **cusp form** for B. The space of cusp forms is denoted $S_2^B(\mathfrak{N})$.

As in the previous cases, there are Hecke operators which act on this space. Let $\mathfrak{p}$ be a nonzero prime ideal in $\mathcal{O}_K$ with $\mathfrak{p} \nmid \mathfrak{N}$. For a right R-ideal I with norm coprime to $\mathfrak{p}$, the action of the Hecke operator $T_\mathfrak{p}$ is defined to be

$$(T_\mathfrak{p} f)([I]) := \sum_{J \subseteq I, N(JI^{-1}) = \mathfrak{p}} f([J]).$$

Theorem 1.8.1. *(Eichler-Shimizu-Jacquet-Langlands) Let B be a quaternion algebra over F of discriminant $\mathfrak{D}$, let $\mathfrak{N}$ be an ideal coprime to $\mathfrak{D}$ and let χ be a character of conductor $\mathfrak{N}$. Then there is an injective map of Hecke modules*

$$S_2^B(\mathfrak{N}, \chi) \quad \hookrightarrow \quad S_2(\mathfrak{D}\mathfrak{N}, \chi)$$

whose image consists of those Hilbert cusp forms which are new at all primes $\mathfrak{p}|\mathfrak{D}$.

We note that in the case that the quaternion algebra B has discriminant $\mathfrak{D} = (1)$, the injection is actually an isomorphism.

Proof. See [22, Section 2.3.6] which in turn gives detailed explanations on references for the proof. □

1.9 Outline of Dembélé's Algorithm

Fix a quaternion algebra B which ramifies at all infinite places of a totally real number field K with ring of integers $\mathcal{O}_K$. Suppose additionally that K has narrow class number one. Let $\mathfrak{D}$ denote the discriminant of the quaternion algebra and let $\mathfrak{N}$ be an ideal coprime to $\mathfrak{D}$. Set $\mathfrak{m} = \mathfrak{D} \cdot \mathfrak{N}$. Then by the Eichler-Shimizu-Jacquet-Langlands correspondence there is an injection of Hecke modules

$$S_2^B(\mathfrak{N}, \chi) \quad \hookrightarrow \quad S_2(\mathfrak{m}, \chi)$$

where the image consists of Hilbert cusp forms new at all primes $\mathfrak{p}|\mathfrak{D}$.

Dembélé provides the following algorithm for computing coefficients of Fourier expansions of such forms (see [10], [11]):

(1) Find a maximal order $R = \mathcal{O}_K e_1 + \mathcal{O}_K e_2 + \mathcal{O}_K e_3 + \mathcal{O}_K e_4$ of B. Compute its group of norm 1 elements R_1. Note that by the assumption that B is ramified at all infinite places, this set will be finite.

(2) Let $\mathfrak{p}$ be a prime in F with totally-positive generator $\pi_\mathfrak{p}$. Find the quaternions in R with norm $\pi_\mathfrak{p}$. Store $\Theta(\mathfrak{p}) := \{x \in R : N(x) = \pi_\mathfrak{p}\}/\sim$, where $x \sim y \leftrightarrow x = y \cdot u, u \in R_1$.

(3) For every prime $\mathfrak{q}|\mathfrak{N}$, find a local isomorphism

$$R \otimes (\mathcal{O}_K/\mathfrak{q}^{e_\mathfrak{q}}) \cong M_2(\mathcal{O}_K/\mathfrak{q}^{e_\mathfrak{q}}).$$

Note that this isomorphism exists as we have chosen $\mathfrak{N}$ coprime to the discriminant $\mathfrak{D}$ of B.

(4) Compute the space $\mathbb{P}^1_\chi(\mathcal{O}_K/\mathfrak{N})$, which is defined to be

$$\mathbb{P}^1_\chi(\mathcal{O}_K/\mathfrak{N}) := \mathcal{H}_1(\mathfrak{N})/\sim$$

where

$$\mathcal{H}_1(\mathfrak{N}) := \left\{(a:b) \in (\mathcal{O}_K/\mathfrak{N}\mathcal{O}_K)^2, \gcd(a,b) \in (\mathcal{O}_K/\mathfrak{N}\mathcal{O}_K)^\times\right\}$$

and where $(a:b) \sim (c:d) \leftrightarrow a = cu, b = du$ for $u \in \ker(\chi)$. By the Chinese Remainder Theorem, this is equivalent to working over the product

$$\mathbb{P}^1_\chi(\mathcal{O}_K/\mathfrak{N}) \cong \prod_{\mathfrak{q}|\mathfrak{N}} \mathbb{P}^1_\chi(\mathcal{O}_K/\mathfrak{q}^{\mathrm{ord}_\mathfrak{q}(\mathfrak{N})}).$$

Let $\mathcal{F}$ denote the fundamental domain, which is the collection of orbits of elements of $\mathbb{P}^1_\chi(\mathcal{O}_K/\mathfrak{m})$ under left-multiplication by an element of R_1.

(5) Compute the action of the Hecke operator $T_\mathfrak{p}$ as follows: for $f \in S_2^B(\mathfrak{N})$, let

$$f||T_\mathfrak{p}(x) = \sum_{u \in \Theta(\mathfrak{p})} u \cdot x, x \in \mathcal{F},$$

where

$$u \cdot x = u \cdot (x_1 : x_2) := (ax_1 + bx_2 : cx_1 + dx_2), u = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

1.10 Examples

To highlight the steps of Dembélé's algorithm, we provide the following examples over the totally real number field $\mathbb{Q}$. That is, we will provide bases for the following spaces of cusp forms:

- $S_2(\Gamma_0(14))$
- $S_2(\Gamma_0(26))$
- $S_2(\Gamma_0(26), \chi)$, for χ a quadratic character of conductor 13.

by considering $B = \left(\frac{-1,-1}{\mathbb{Q}}\right)$ which has discriminant $\mathfrak{D} = 2$ (meaning it ramifies at the only infinite place and at $p = 2$). Let R denote the maximal order of B given by

$$R = \mathbb{Z}e_1 + \mathbb{Z}e_2 + \mathbb{Z}e_3 + \mathbb{Z}e_4$$

where

$$\begin{aligned} e_1 &= \frac{1}{2}(1 + i + j + k) \\ e_2 &= i \\ e_3 &= j \\ e_4 &= k. \end{aligned}$$

Given an element $x = Ae_1 + Be_2 + Ce_3 + De_4 = (A, B, C, D) \in R$, the reduced norm $N(x) = A^2 + AB + AC + AD + B^2 + C^2 + D^2$ is a positive definite quadratic form.

We begin by finding the elements $u \in R_1$:

$$\begin{aligned} R_1 = \pm\{&(2,-1,-1,-1),(1,0,0,0),(1,0,0,-1),(1,0,-1,0),(1,0,-1,-1),(1,-1,0,0), \\ &(1,-1,0,-1),(1,-1,-1,0),(1,-1,-1,-1),(0,1,0,0),(0,0,1,0),(0,0,0,1)\}. \end{aligned}$$

Next we compute $\Theta(p)$ for various p. This was accomplished by the bounding region code which will be discussed at length in the next chapter.

- $p = 2$. The elements $x = (A, B, C, D) \in R$ of norm 2 are:

$$\pm \{(0,0,1,-1),(0,0,1,1),(0,1,-1,0),(0,1,0,-1),(0,1,0,1),(0,1,1,0),$$
$$(2,-2,-1,-1),(2,-1,-2,-1),(2,-1,-1,-2),(2,-1,-1,0),(2,-1,0,-1),(2,0,-1,-1)\}.$$

  Taking equivalences up to units, we see

$$\Theta(2) = \{(0,0,1,1)\} \leftrightarrow \{j+k\}.$$

- $p = 3$. The elements $x = (A, B, C, D) \in R$ of norm 3 are:

$$\pm \{(3,-1,-1,-1),(3,-2,-1,-1),(2,0,0,-1),(2,-1,0,0)$$
$$(3,-1,-1,-2),(3,-2,-1,-2),(2,0,-1,0),(2,-1,0,-2)$$
$$(3,-1,-2,-1),(3,-2,-2,-1),(2,0,-1,-2),(2,-1,-2,0)$$
$$(3,-1,-2,-2),(3,-2,-2,-2),(2,0,-2,-1),(2,-1,-2,-2)$$
$$(2,-2,0,-1),(1,1,0,0),(1,0,1,0),(1,0,-1,1)$$
$$(2,-2,-1,0),(1,1,0,-1),(1,0,1,-1),(1,0,-1,-2)$$
$$(2,-2,-1,-2),(1,1,-1,0),(1,0,0,1),(1,0,-2,0)$$
$$(2,-2,-2,-1),(1,1,-1,-1),(1,0,0,-2),(1,0,-2,-1)$$
$$(1,-1,1,0),(1,-1,-1,1),(1,-2,0,0),(0,1,1,1)$$
$$(1,-1,1,-1),(1,-1,-1,-2),(1,-2,0,-1),(0,1,1,-1)$$
$$(1,-1,0,1),(1,-1,-2,0),(1,-2,-1,0),(0,1,-1,1)$$
$$(1,-1,0,-2),(1,-1,-2,-1),(1,-2,-1,-1),(0,1,-1,-1)\}.$$

Taking equivalences up to units, then,

$$\begin{aligned}\Theta(3) &= \{(0,1,1,1),(0,1,1,-1),(0,1,-1,1),(0,1,-1,-1)\} \\ &= \{i+j+k, i+j-k, i-j+k, i-j-k\}.\end{aligned}$$

Example 1.10.1. *We compute the first few Fourier coefficients of the cusp forms which provide a Hecke eigenbasis for* $S_2(14,\mathbb{1})$.

Note that in this case we have $\mathfrak{N}=7$, $\mathfrak{m}=14$.

We take the following local isomorphism $R\otimes(\mathbb{Z}/7\mathbb{Z})\cong M_2(\mathbb{Z}/7\mathbb{Z})$:

$$1\mapsto\begin{pmatrix}1&0\\0&1\end{pmatrix}\qquad i\mapsto\begin{pmatrix}0&1\\-1&0\end{pmatrix}$$
$$j\mapsto\begin{pmatrix}2&3\\3&5\end{pmatrix}\qquad k\mapsto\begin{pmatrix}3&5\\-2&-3\end{pmatrix}$$

Noting that

$$\mathbb{P}^1_{\mathbb{1}}(\mathbb{Z}/7\mathbb{Z})=\mathbb{P}^1(\mathbb{Z}/7\mathbb{Z})=\{(1:0),(0:1),(1:1),(2:1),(3:1),(4:1),(5:1),(6:1)\}$$

we next consider the image of $R_1/\{\pm1\}$ under the local isomorphism:

$$(0,0,0,1)\leftrightarrow\begin{pmatrix}3&5\\-2&-3\end{pmatrix}\qquad(0,0,1,0)\leftrightarrow\begin{pmatrix}2&3\\3&5\end{pmatrix}$$

$$(0,1,0,0)\leftrightarrow\begin{pmatrix}0&1\\-1&0\end{pmatrix}\qquad(2,-1,-1,-1)\leftrightarrow\begin{pmatrix}1&0\\0&1\end{pmatrix}$$

$$(1,-1,-1,-1) \leftrightarrow \begin{pmatrix} 5 & 6 \\ 0 & 3 \end{pmatrix} \qquad (1,0,0,0) \leftrightarrow \begin{pmatrix} 3 & 1 \\ 0 & 5 \end{pmatrix}$$

$$(1,0,0,-1) \leftrightarrow \begin{pmatrix} 0 & 3 \\ 2 & 1 \end{pmatrix} \qquad (1,0,-1,0) \leftrightarrow \begin{pmatrix} 1 & 5 \\ 4 & 0 \end{pmatrix}$$

$$(1,-1,0,0) \leftrightarrow \begin{pmatrix} 3 & 0 \\ 1 & 5 \end{pmatrix} \qquad (1,0,-1,-1) \leftrightarrow \begin{pmatrix} 5 & 0 \\ 6 & 3 \end{pmatrix}$$

$$(1,-1,0,-1) \leftrightarrow \begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix} \qquad (1,-1,-1,0) \leftrightarrow \begin{pmatrix} 1 & 4 \\ 5 & 0 \end{pmatrix}.$$

This gives a fundamental domain of

$$\mathcal{F} = \{\{(1:0),(0:1),(2:1),(3:1)\},\{(1:1),(4:1),(5:1),(6:1)\}\}.$$

The Hecke action then is as follows:

- For $p = 2$, we have

$$j + k \leftrightarrow \begin{pmatrix} 5 & 8 \\ 1 & 2 \end{pmatrix}$$

$$\begin{aligned} f_0||T_2(1:0) &= \begin{pmatrix} 5 & 8 \\ 1 & 2 \end{pmatrix}(1:0) \\ &= (5:1) \\ f_1||T_2(1:1) &= \begin{pmatrix} 5 & 8 \\ 1 & 2 \end{pmatrix}(1:1) \\ &= (13:3) = (2:1). \end{aligned}$$

Thus, $T_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. The eigenvalues are $\lambda_1 = -\lambda_2 = 1$. The eigenvector corresponding to λ_1 is $(1,1)$ (our Eisenstein component), and the eigenvector corresponding to $\lambda_2 = 1$ is $(1,-1)$ (our cusp).

- For $p = 3$ we have

$$i+j+k \leftrightarrow \begin{pmatrix} 5 & 9 \\ 0 & 2 \end{pmatrix} \qquad i+j-k \leftrightarrow \begin{pmatrix} -1 & -1 \\ 4 & 2 \end{pmatrix}$$
$$i-j+k \leftrightarrow \begin{pmatrix} 1 & 3 \\ -6 & 2 \end{pmatrix} \qquad i-j-k \leftrightarrow \begin{pmatrix} -5 & -7 \\ 2 & -2 \end{pmatrix}.$$

Then:

$$\begin{aligned} f_0||T_3(1:0) &= \begin{pmatrix} 5 & 9 \\ 0 & 2 \end{pmatrix}(1:0) + \begin{pmatrix} -1 & -1 \\ 4 & 2 \end{pmatrix}(1:0) \\ &\quad + \begin{pmatrix} 1 & 3 \\ -6 & 2 \end{pmatrix}(1:0) + \begin{pmatrix} -5 & -7 \\ 2 & -2 \end{pmatrix}(1:0) \\ &= (5:0) + (-1:4) + (1:-6) + (-5:2) \\ &= (1:0) + (5:1) + (1:1) + (1:1) \end{aligned}$$

$$\begin{aligned} f_1||T_3(1:1) &= \begin{pmatrix} 5 & 9 \\ 0 & 2 \end{pmatrix}(1:1) + \begin{pmatrix} -1 & -1 \\ 4 & 2 \end{pmatrix}(1:1) \\ &\quad + \begin{pmatrix} 1 & 3 \\ -6 & 2 \end{pmatrix}(1:1) + \begin{pmatrix} -5 & -7 \\ 2 & -2 \end{pmatrix}(1:1) \\ &= (0:1) + (2:1) + (1:1) + (1:0). \end{aligned}$$

Thus, $T_3 = \begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix}$. The eigenvalues are $\lambda_1 = 4$ and $\lambda_2 = -2$. For $\lambda_1 = 4$, the corresponding eigenvector is $(1, 1)$ (the Eisenstein component). For $\lambda_2 = -2$, the corresponding eigenvector is $(1, -1)$ (the cusp component).

Thus the cusp space $S_2(14, \mathbb{1})$ is one-dimensional with basis vector $q - q^2 - 2q^3 + ...$

Example 1.10.2. *We compute the first few Fourier coefficients of the cusp forms which provide a Hecke eigenbasis for* $S_2(26, \mathbb{1})$.

Here we have $\mathfrak{N} = 13, \mathfrak{m} = 26$. For our local isomorphism $R \otimes (\mathbb{Z}/13\mathbb{Z}) \cong M_2(\mathbb{Z}/13\mathbb{Z})$ we choose:

$$1 \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad i \mapsto \begin{pmatrix} 0 & 1 \\ 12 & 0 \end{pmatrix}$$
$$j \mapsto \begin{pmatrix} 3 & 4 \\ 4 & 10 \end{pmatrix} \qquad k \mapsto \begin{pmatrix} 4 & 10 \\ 10 & 9 \end{pmatrix}.$$

Noting that

$$\mathbb{P}^1_{\mathbb{1}}(\mathbb{Z}/13\mathbb{Z}) = \mathbb{P}^1(\mathbb{Z}/13\mathbb{Z}) = \{(1:0), (0:1), (1:1), (2:1), (3:1), (4:1), (5:1), (6:1),$$
$$(7:1), (8:1), (9:1), (10:1), (11:1), (12:1)\}$$

we next consider the image of $R_1/\{\pm 1\}$ under the local isomorphism:

$$(0,0,0,1) \leftrightarrow \begin{pmatrix} 4 & 10 \\ 10 & 9 \end{pmatrix} \qquad (0,0,1,0) \leftrightarrow \begin{pmatrix} 3 & 4 \\ 4 & 10 \end{pmatrix}$$

$$(0,1,0,0) \leftrightarrow \begin{pmatrix} 0 & 1 \\ 12 & 0 \end{pmatrix} \qquad (2,-1,-1,-1) \leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$(1,-1,-1,-1) \leftrightarrow \begin{pmatrix} 10 & 12 \\ 0 & 4 \end{pmatrix} \qquad (1,0,0,0) \leftrightarrow \begin{pmatrix} 4 & 1 \\ 0 & 10 \end{pmatrix}$$

$$(1,0,0,-1) \leftrightarrow \begin{pmatrix} 0 & 4 \\ 3 & 1 \end{pmatrix} \qquad (1,0,-1,0) \leftrightarrow \begin{pmatrix} 1 & 10 \\ 9 & 0 \end{pmatrix}$$

$$(1,-1,0,0) \leftrightarrow \begin{pmatrix} 4 & 0 \\ 1 & 10 \end{pmatrix} \qquad (1,0,-1,-1) \leftrightarrow \begin{pmatrix} 10 & 0 \\ 12 & 4 \end{pmatrix}$$

$$(1,-1,0,-1) \leftrightarrow \begin{pmatrix} 0 & 3 \\ 4 & 1 \end{pmatrix} \qquad (1,-1,-1,0) \leftrightarrow \begin{pmatrix} 1 & 9 \\ 10 & 0 \end{pmatrix}.$$

The fundamental domain is therefore

$$\mathcal{F} = \{\{(1:0),(0:1),(3:1),(4:1)\},\{(1:1),(7:1),(11:1),(12:1)\},$$
$$\{(2:1),(5:1),(6:1),(8:1),(9:1),(10:1)\}\}.$$

The Hecke action then is as follows:

- For $p = 2$

$$\begin{aligned} j+k &\Rightarrow \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix} \\ f_0||T_2(1:0) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(1:0) \\ &= (7:1) \\ f_1||T_2(1:1) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(1:1) \\ &= (8:7) = (3:1) \\ f_2||T_2(2:1) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(2:1) \\ &= (15:8) = (10:1). \end{aligned}$$

So $T_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, which has eigenvalues $\lambda_1 = -1$, $\lambda_2 = \lambda_3 = 1$. For $\lambda = -1$ we have eigenvector $(1, -1, 0)$, and for the eigenvalues $\lambda_2 = \lambda_3 = 1$ we have eigenvectors $(1, 1, 0)$ and $(0, 0, 1)$.

- For $p = 3$

$$i+j+k \leftrightarrow \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix} \qquad i+j-k \leftrightarrow \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}$$

$$i-j+k \leftrightarrow \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix} \qquad i-j-k \leftrightarrow \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}$$

and

$$\begin{aligned} f_0||T_3(1:0) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(1:0) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(1:0) \\ &\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(1:0) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(1:0) \\ &= (7:0) + (12:6) + (1:5) + (6:11) \\ &= (1:0) + (2:1) + (8:1) + (10:1) \end{aligned}$$

$$\begin{aligned} f_1||T_3(1:1) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(1:1) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(1:1) \\ &\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(1:1) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(1:1) \\ &= (9:6) + (20:7) + (8:17) + (6:18) \\ &= (8:1) + (1:1) + (2:1) + (9:1) \end{aligned}$$

$$\begin{aligned} f_2||T_3(2:1) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(2:1) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(2:1) \\ &\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(2:1) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(2:1) \\ &= (16:6) + (32:13) + (9:22) + (12:29) \\ &= (7:1) + (1:0) + (1:1) + (4:1). \end{aligned}$$

Thus $T_3 = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 3 \\ 2 & 2 & 0 \end{pmatrix}$ with eigenvectors $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = -3$. The eigenvector associated to $\lambda_1 = 4$ is our Eisenstein component $(1, 1, 1)$. For $\lambda_2 = 1$ we have vector $(1, -1, 0)$, and for vector $\lambda_3 = -3$, we have $(1, 1, -4/3)$.

The cusp space $S_2(26, \mathbb{1})$ is therefore two-dimensional with spanning cusp forms:

$$q + q^2 - 3q^3 + ...$$
$$q - q^2 + q^3 + ...$$

Example 1.10.3. *We compute the first few Fourier coefficients of the cusp forms which give a Hecke eigen basis for the space $S_2(26, \chi)$, where χ is the character of conductor 13 sending $15 \mapsto -1$.*

Note how the character χ behaves on primes p:

$$\begin{aligned} \chi(p) &= \begin{cases} 1, & p \equiv 1, 3, 9, 17, 23, 25 \pmod{26} \\ -1, & n \equiv 5, 7, 11, 15, 19, 21 \pmod{26} \end{cases} \\ &= \begin{cases} 1, & p \equiv 1, 3, 4, 9, 10, 12 \pmod{13} \\ -1, & n \equiv 2, 5, 6, 7, 8, 11 \pmod{13}. \end{cases} \end{aligned}$$

Therefore, we claim that

$$\begin{aligned} \mathbb{P}^1_\chi(\mathbb{Z}/13\mathbb{Z}) = \{&(0:1), (1:1), (2:1), (3:1), (4:1), (5:1), (6:1), (7:1), (8:1), \\ &(9:1), (10:1), (11:1), (12:1), (1:0), (2:0), (0:2), (1:2), (2:2), \\ &(3:2), (4:2), (5:2), (6:2), (7:2), (8:2), (9:2), (10:2), (11:2), (12:2)\}. \end{aligned}$$

Considering the action by R_1, we obtain 5 orbits for the fundamental domain $\mathcal{F}$:

$$\{\{(0:1),(1:0),(3:1),(4:1)\},\{(1:1).(7:1),(12:1),(9:2)\},$$
$$\{(2:0),(0:2),(6:2),(8:2)\},\{(1:2),(11:1),(11:2),(2:2)\},$$
$$\{(2:1),(12:2),(4:2),(6:1),(8:1),(10:1),(3:2),(10:2),(5:1),(7:2),(9:1),(5:2)\}\}.$$

In computing the action of the Hecke operators, we find:

- When $p = 2$:

$$\begin{aligned} f_0||T_2(0:1) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(0:1) \\ &= (1:6) = (9:2) \end{aligned}$$

$$\begin{aligned} f_1||T_2(1:1) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(1:1) \\ &= (8:7) = (6:2) \end{aligned}$$

$$\begin{aligned} f_2||T_2(2:0) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(2:0) \\ &= (14:2) = (1:2) \end{aligned}$$

$$\begin{aligned} f_3||T_2(1:2) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(1:2) \\ &= (9:13) = (1:0) \end{aligned}$$

$$\begin{aligned} f_4||T_2(2:1) &= \begin{pmatrix} 7 & 1 \\ 1 & 6 \end{pmatrix}(2:1) \\ &= (15:8) = (7:2). \end{aligned}$$

So $B_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$. The eigenspace information can be found below (and was computed using Sage):

```
[(-1, Vector space of degree 5 and dimension 1 over Rational Field
User basis matrix:
[ 1 -1  1 -1  0]),
(1, Vector space of degree 5 and dimension 2 over Rational Field
User basis matrix:
[1 1 1 1 0]
[0 0 0 0 1]),
(-1*I, Vector space of degree 5 and dimension 1 over Algebraic Field
User basis matrix:
[   1 -1*I   -1  1*I    0]),
(1*I, Vector space of degree 5 and dimension 1 over Algebraic Field
User basis matrix:
[   1  1*I   -1 -1*I    0])]
```

- When $p = 3$:

$$\begin{aligned} f_0||T_3(0:1) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(0:1) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(0:1) \\ &\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(0:1) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(0:1) \\ &= (2:6) + (8:1) + (7:12) + (0:7) \\ &= (5:2) + (8:1) + (6:1) + (0:2) \end{aligned}$$

$$\begin{aligned} f_1||T_3(1:1) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(1:1) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(1:1) \\ &\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(1:1) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(1:1) \\ &= (9:6) + (20:7) + (8:17) + (6:18) \\ &= (3:2) + (2:2) + (2:1) + (5:2) \end{aligned}$$

$$\begin{aligned} f_2||T_3(2:0) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(2:0) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(2:0) \\ &\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(2:0) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(2:0) \\ &= (14:0) + (24:12) + (2:10) + (12:22) \\ &= (1:0) + (2:1) + (8:1) + (10:1) \end{aligned}$$

$$
\begin{aligned}
f_3||T_3(1:2) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(1:2) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(1:2) \\
&\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(1:2) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(1:2) \\
&= (11:12) + (20:8) + (15:29) + (6:28) \\
&= (2:1) + (5:2) + (5:1) + (7:1)
\end{aligned}
$$

$$
\begin{aligned}
f_4||T_3(2:1) &= \begin{pmatrix} 7 & 2 \\ 0 & 6 \end{pmatrix}(2:1) + \begin{pmatrix} 12 & 8 \\ 6 & 1 \end{pmatrix}(2:1) \\
&\quad + \begin{pmatrix} 1 & 7 \\ 5 & 12 \end{pmatrix}(2:1) + \begin{pmatrix} 6 & 0 \\ 11 & 7 \end{pmatrix}(2:1) \\
&= (16:6) + (32:13) + (9:22) + (12:29) \\
&= (1:2) + (2:0) + (1:1) + (4:1).
\end{aligned}
$$

This means $B_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 3 \\ 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & 3 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$. The eigenspaces are listed below:

```
[(4, Vector space of degree 5 and dimension 1 over Rational Field
User basis matrix:
[1 1 1 1 1]),
(1, Vector space of degree 5 and dimension 1 over Rational Field
User basis matrix:
[ 1 -1  1 -1  0]),
(-3, Vector space of degree 5 and dimension 1 over Rational Field
```

```
User basis matrix:
[   1    1    1    1 -4/3]),
(-1, Vector space of degree 5 and dimension 2 over Rational Field
User basis matrix:
[ 1  0 -1  0  0]
[ 0  1  0 -1  0])]
```

The cusp space $S_2(26, \chi)$ is therefore two-dimensional with spanning cusp forms:

$$q - iq^2 - q^3 + ...$$
$$q + iq^2 - q^3 + ...$$

Chapter 2

Bounding Regions

2.1 Summary

This chapter outlines the algorithms which were coded for this project: for Q a quaternary totally positive definite integral form over $K = \mathbb{Q}(\sqrt{d})$ $(d > 0)$ and m a totally positive integer of K, we return $r_Q(m) := \#\{\vec{x} \in (\mathcal{O}_K)^4 : Q(\vec{x}) = m\}$. This is accomplished by first diagonalizing the form over K, and then using the change of basis matrix to recursively bound each of the four variables. Once all the "bounded regions" have been created, we enumerate over the sets to recover multiple $r_Q(m)$ values. In addition to proving the algorithms behind the code, we provide examples of outputs (some of which will be relevant to later chapters).

2.2 Preliminaries

Let $K = \mathbb{Q}(\sqrt{d})$ be a real quadratic number field with ring of integers $\mathcal{O}_K = \{a + b\theta : a, b \in \mathbb{Z}\}$, where

$$\theta = \begin{cases} \sqrt{d} & d \equiv 2, 3 \pmod 4 \\ \frac{1+\sqrt{d}}{2} & d \equiv 1 \pmod 4. \end{cases}$$

There are two embeddings of K into $\mathbb{R}$:

$$\begin{aligned} id : K &\hookrightarrow \mathbb{R} \\ x + y\theta &\mapsto x + y\theta \end{aligned}$$

$$\begin{aligned} \bar{\cdot} : K &\hookrightarrow \mathbb{R} \\ x + y\theta &\mapsto x + y\overline{\theta} \end{aligned}$$

where

$$\overline{\theta} = \begin{cases} -\sqrt{d} & d \equiv 2, 3 \pmod 4 \\ \frac{1-\sqrt{d}}{2} & d \equiv 1 \pmod 4. \end{cases}$$

We then identify $\mathcal{O}_K$ as a discrete subset of $\mathbb{R}^2$ by the following:

$$\begin{aligned} \mathcal{O}_K &\hookrightarrow \mathbb{R}^2 \\ \alpha &\mapsto (id(\alpha), \overline{\alpha}). \end{aligned}$$

Example 2.2.1. *The lattice associated to* $\mathbb{Q}(\sqrt{5})$ *(note that the totally positive elements are in the first quadrant):*

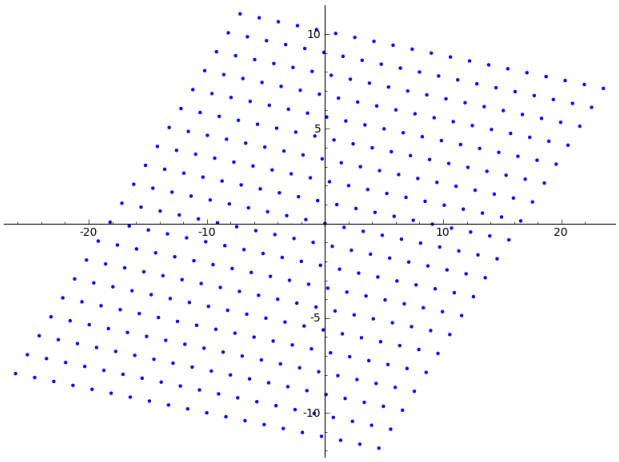

Figure 2.1: The lattice for $\mathbb{Q}(\sqrt{5})$

Theorem 2.2.1. *In a real quadratic number field $K = \mathbb{Q}(\sqrt{d})$, there are finitely many totally positive integers of fixed trace.*

Proof. Case (I): $d \equiv 2, 3 \pmod 4$. Given $\alpha = i + j\theta$, $(i, j \in \mathbb{Z})$, α totally positive forces $i > 0$. Since $Tr(\alpha) = 2i$, for each possible trace value there is exactly one possible i value. Then to ensure total-positivity:

$$\begin{aligned} i + j\theta &> 0 \\ j &> -i/\theta \\ i - j\theta &> 0 \\ i/\theta &> j. \end{aligned}$$

Therefore, $j \in (-i/\theta, i/\theta) \cap \mathbb{Z}$, and the statement holds.

Case (II): $d \equiv 1 \pmod 4$. Given $\alpha = i + j\theta$, $(i, j \in \mathbb{Z})$, α totally positive again forces $i > 0$; however, now $Tr(\alpha) = 2i + j = M \Rightarrow j = M - 2i$. So, for each admissible i-value, there is exactly one j-value. Moreover,

$$\begin{aligned} i + j(1 + \sqrt{d})/2 &> 0 \\ j &> (-2i)/(1 + \sqrt{d}) \\ i + j(1 - \sqrt{d})/2 &> 0 \\ j &< (2i)/(\sqrt{d} - 1) \end{aligned}$$

implies

$$\begin{array}{ccc} \dfrac{-2i}{1+\sqrt{d}} & < M-2i < & \dfrac{2i}{\sqrt{d}-1} \\ \dfrac{2i\sqrt{d}}{1+\sqrt{d}} & < M < & \dfrac{2i\sqrt{d}}{\sqrt{d}-1} \\ & \Rightarrow & \\ M\left(\dfrac{\sqrt{d}-1}{2\sqrt{d}}\right) & < i < & M\left(\dfrac{1+\sqrt{d}}{2\sqrt{d}}\right). \end{array}$$

Thus for each M, there are finitely-many possible i-values, each of which has a unique j-value. Hence the result holds. □

N.B. This statement is actually trivialized by the geometric representation given earlier. For a given trace value T, consider the line $y = -x + T$. Intersecting this line with our lattice yields finitely many points in the first quadrant. Consider the example below of the lattice associated to $\mathbb{Q}(\sqrt{5})$ along with the line $y = -x + 5$. All points bounded by this line and the first quadrant are the totally positive elements of $\mathcal{O}_{\mathbb{Q}(\sqrt{5})}$ with trace at most 5.

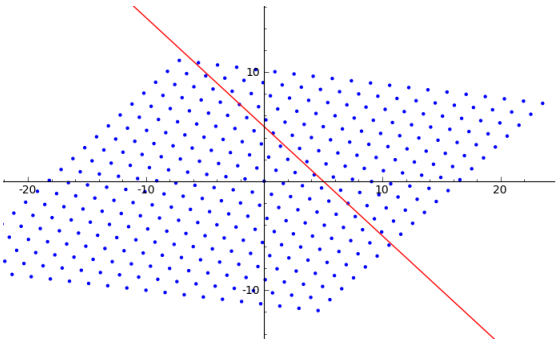

Figure 2.2: Elements of trace at most 5 in $\mathbb{Q}(\sqrt{5})$

2.3 Creating Regions

Note: The following algorithm holds for an arbitrary number of variables. We outline it for the quaternary case.

Consider a totally positive definite, quaternary integral quadratic form Q over K. Suppose for some $m \in \mathcal{O}_K^+$, we want to determine

$$r_Q(m) := \#\{\vec{x} \in \mathcal{O}_K^4 | Q(\vec{x}) = m\}.$$

It would suffice to determine a finite set $S \subset \mathcal{O}_K^4$ containing all $\vec{x}$ such that $Q(\vec{x}) = m$.

We begin by diagonalizing Q over K to obtain a similar form $\widetilde{Q}$; i.e.,

$$Q(\vec{x}) \sim \widetilde{Q}(\vec{y}) = c_{11}y_1^2 + c_{22}y_2^2 + c_{33}y_3^2 + c_{44}y_4^2,$$

where for all i $c_{ii} \in K^+$. By the Gram-Schmidt process, we can ensure

$$\begin{aligned} y_4 &= x_4 \\ y_3 &= x_3 + a_{34}x_4 \\ y_2 &= x_2 + a_{23}x_3 + a_{24}x_4 \\ y_1 &= x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 \end{aligned}$$

with $a_{ij} \in K$. We proceed by creating bounded regions for the y_i's which using the change of basis map we will then use to obtain bounded regions for the x_i's. We begin by bounding $y_4 = x_4$ and build successive regions recursively.

As $y_4 = x_4$:

$$\begin{aligned} m &\geq c_{11}y_1^2 + c_{22}y_2^2 + c_{33}y_3^2 + c_{44}x_4^2 \\ (m/c_{44}) &\geq (c_{11}/c_{44})y_1^2 + (c_{22}/c_{44})y_2^2 + (c_{33}/c_{44})c_3^2 + x_4^2 \\ \overline{(m/c_{44})} &\geq \overline{(c_{11}/c_{44})}\overline{y_1}^2 + \overline{(c_{22}/c_{44})}\overline{y_2}^2 + \overline{(c_{33}/c_{44})}\overline{y_3}^2 + \overline{x_4}^2 \\ Tr\left(\frac{m}{c_{44}}\right) &\geq \underbrace{Tr\left(\sum_{i\neq 4}\frac{c_{ii}}{c_{44}}y_i^2\right)}_{*} + \left(x_4^2 + \overline{x_4}^2\right). \end{aligned}$$

Since by total positive-definiteness, $* \geq 0$, we simplify this to

$$x_4^2 + \overline{x_4}^2 \leq Tr\left(\frac{m}{c_{44}}\right)$$

and conclude that all candidates for x_4 lie on or inside the circle of radius $\sqrt{B_4} := \sqrt{Tr(m/c_{44})}$ centered about the origin. Explicitly, writing $x_4 = i_4 + j_4\theta$, for $i_4, j_4 \in \mathbb{Z}$, we have

$$x_4^2 + \overline{x_4}^2 = \begin{cases} 2i_4^2 + 2dj_4^2 & d \equiv 2,3 \pmod 4 \\ i_4^2 + (i_4 + j_4)^2 + \left(\frac{d-1}{2}\right) j_4^2 & d \equiv 1 \pmod 4. \end{cases}$$

Thus $x_4^2 + \overline{x_4}^2 \leq B_4$ implies

$$\begin{aligned} i_4 \in \left(-\sqrt{B_4/2}, \sqrt{B_4/2}\right) \qquad & j_4 \in \left(-\sqrt{B_4/2d}, \sqrt{B_4/2d}\right) \qquad d \equiv 2,3 \pmod 4 \\ i_4 \in \left(-\sqrt{B_4}, \sqrt{B_4}\right) \quad & j_4 \in \left(-\sqrt{2B_4/(d-1)}, \sqrt{2B_4/(d-1)}\right) \quad d \equiv 1 \pmod 4. \end{aligned}$$

We now have explicit bounded regions in which i_4 and j_4 must lie, which we use to obtain similar regions for x_3:

Fix a particular $x_4 = i_4 + j_4\theta$ from the previous step. We know

$$y_3^2 + \overline{y_3}^2 \leq \underbrace{Tr\left(\frac{m}{c_{33}}\right)}_{B_3}.$$

This trivially gives

$$\begin{aligned} -\sqrt{B_3} &\leq y_3 \leq \sqrt{B_3} \\ -\sqrt{B_3} &\leq \overline{y_3} \leq \sqrt{B_3}, \end{aligned}$$

which in turn means

$$\begin{aligned} -\sqrt{B_3} &\leq x_3 + a_{34}x_4 \leq \sqrt{B_3} \\ -\sqrt{B_3} &\leq \overline{x_3} + \overline{a_{34}x_4} \leq \sqrt{B_3}. \end{aligned}$$

Expanding these equations yields:

$$\begin{aligned} -\sqrt{B_3} - a_{34}(i_4 + j_4\theta) &\leq i_3 + j_3\theta \leq \sqrt{B_3} - a_{34}(i_4 + j_4\theta) \\ -\sqrt{B_3} - \overline{a_{34}}(i_4 + j_4\overline{\theta}) &\leq i_3 + j_3\overline{\theta} \leq \sqrt{B_3} - \overline{a_{34}}(i_4 + j_4\overline{\theta}). \end{aligned}$$

Let $K_3 := a_{34}(i_4 + j_4\theta)$. When $d \equiv 1 \pmod 4$, we see

$$\frac{1}{\sqrt{d}}\left(-2\sqrt{B_3} - K_3 + \overline{K_3}\right) \leq j_3 \leq \frac{1}{\sqrt{d}}\left(2\sqrt{B_3} - K_3 + \overline{K_3}\right).$$

We now have two ways we can proceed:

- We can define i_3 to be dependent upon j_3. To do this we note that upon bounding j_3, we have:

$$\frac{1}{2}\left(-2\sqrt{B_3} - Tr\left(K_3\right) - j_3\right) \leq i_3 \leq \frac{1}{2}\left(2\sqrt{B_3} - Tr\left(K_3\right) - j_3\right).$$

- If we do not want i_3 to be dependent upon j_3, we can proceed as follows:

$$\begin{aligned}
\overline{\theta}(\sqrt{B_3}+K_3) &\leq -\overline{\theta}(i_3+j_3\theta) \leq & -\overline{\theta}(\sqrt{B_3}-K_3)\\
-\theta(\sqrt{B_3}+\overline{K_3}) &\leq \theta(i_3+j_3\theta) \leq & \theta(\sqrt{B_3}-\overline{K_3})\\
(\theta-\overline{\theta})(-\sqrt{B_3})+\overline{\theta}K_3-\theta\overline{K_3} &\leq (\theta-\overline{\theta})i_3 \leq & (\theta-\overline{\theta})\sqrt{B_3}+\overline{\theta}K_3-\theta\overline{K_3}\\
-\sqrt{B_3}+\frac{1}{\sqrt{d}}\left(\overline{\theta}K_3-\theta\overline{K_3}\right) &\leq i_3 \leq & \sqrt{B_3}+\frac{1}{\sqrt{d}}\left(\overline{\theta}K_3-\theta\overline{K_3}\right).
\end{aligned}$$

N.B. As the independence of i gives a much better bound, that is what is coded.

We continue this recursive process to obtain a set containing all

$$\{x_k = i_k + j_k\theta | Q(\vec{x}) = m\}.$$

Similarly when $d \equiv 2, 3 \pmod 4$,

$$\frac{1}{2}\left(-2\sqrt{B_3}-Tr(K_3)\right) \leq i_3 \leq \frac{1}{2}\left(2\sqrt{B_3}-Tr(K_3)\right),$$

and

$$\frac{1}{2\sqrt{d}}\left(-2\sqrt{B_3}-K_3+\overline{K_3}\right) \leq j_3 \leq \frac{1}{2\sqrt{d}}\left(2\sqrt{B_3}-K_3+\overline{K_3}\right).$$

Again, we continue this recursive process to obtain a set containing all

$$\{x_k = i_k + j_k\theta | Q(\vec{x}) = m\}.$$

For completeness, we describe the generalization of this algorithm:

- Let $B_k = Tr\left(\frac{m}{c_{kk}}\right)$.
- Let $K_k = \sum_{j=k+1}^{n} a_{kj}\,(i_k + j_k\theta)$.

For $d \equiv 1 \pmod 4$:

$$\begin{aligned} \frac{1}{\sqrt{d}}\left(-2\sqrt{B_k} - K_k + \overline{K_k}\right) &\leq j_k \leq \frac{1}{\sqrt{d}}\left(2\sqrt{B_k} - K_k + \overline{K_k}\right) \\ \frac{1}{2}\left(-2\sqrt{B_k} - Tr(K_k) - j_k\right) &\leq i_k \leq \frac{1}{2}\left(2\sqrt{B_k} - Tr(K_k) - j_k\right) \end{aligned}$$

or

$$-\sqrt{B_k} + \frac{1}{\sqrt{d}}\left(\overline{\theta}K_k - \theta\overline{K_k}\right) \leq i_k \leq \sqrt{B_k} + \frac{1}{\sqrt{d}}\left(\overline{\theta}K_k - \theta\overline{K_k}\right)$$

and for $d \equiv 2, 3 \pmod 4$:

$$\begin{aligned} \frac{1}{2}\left(-2\sqrt{B_k} - Tr(K_k)\right) &\leq i_k \leq \frac{1}{2}\left(2\sqrt{B_k} - Tr(K_k)\right) \\ \frac{1}{2\sqrt{d}}\left(-2\sqrt{B_k} - K_k + \overline{K_k}\right) &\leq j_k \leq \frac{1}{2\sqrt{d}}\left(2\sqrt{B_k} - K_k + \overline{K_k}\right). \end{aligned}$$

2.4 Returning Multiple Representation Numbers

The bounding regions for the y_i described above are circles of radius $\sqrt{Tr\left(\frac{m}{c_{ii}}\right)}$ centered at the origin, where $m \in \mathcal{O}_K^+$ and $c_{ii} \in K^+$. We can use these regions to compute $r_Q(\ell)$ for certain $m \neq \ell \in \mathcal{O}_K^+$.

A necessary and sufficient condition on ℓ is that for all i,

$$\sqrt{Tr\left(\frac{\ell}{c_{ii}}\right)} \leq \sqrt{Tr\left(\frac{m}{c_{ii}}\right)}.$$

This means for all i:

$$\begin{aligned} Tr\left(\frac{\ell}{c_{ii}}\right) &\leq Tr\left(\frac{m}{c_{ii}}\right) \\ \frac{\ell}{c_{ii}} + \frac{\overline{\ell}}{\overline{c_{ii}}} &\leq Tr\left(\frac{m}{c_{ii}}\right) \\ \overline{\ell} &\leq \overline{c_{ii}}\left(Tr\left(\frac{m}{c_{ii}}\right)\right) - \left(\frac{\overline{c_{ii}}}{c_{ii}}\right)\ell. \end{aligned}$$

By positive-definiteness, $-\left(\frac{\overline{c_{ii}}}{c_{ii}}\right) < 0$. Thus, for each i, our candidates for ℓ are finite in number and are restricted to the points $(\ell, \overline{\ell}) \in \mathbb{R}^2$ in the first quadrant on or below the line

$$L_i : y = \overline{c_{ii}}\left(Tr\left(\frac{m}{c_{ii}}\right)\right) - \frac{\overline{c_{ii}}}{c_{ii}}x.$$

Let $N_{(m,i)} := \left\{\ell \in \mathcal{O}_K^+ | \overline{\ell} \leq \overline{c_{ii}}\left(Tr\left(\frac{m}{c_{ii}}\right)\right) - \left(\frac{\overline{c_{ii}}}{c_{ii}}\right)\ell\right\}$ and let $N_{(m,Q)} := \bigcap_{i=1}^{4} N_{(m,i)}$. Note that for each i, $m \in N_{(m,i)}$ which implies $N_{(m,Q)} \neq \emptyset$. Moreover, since each $|N_{(m,j)}| < \infty$, $N_{(m,Q)}$ is finite. Most importantly, $N_{(m,Q)}$ is precisely those $\ell \in \mathcal{O}_K^+$ such that the bounding boxes for $r_Q(\ell)$ are contained in the bounding boxes for $r_Q(m)$.

Example 2.4.1. *Suppose $\forall i$, $c_{ii} \in \mathbb{Q}^{>0}$. Then for any m:*

$$Tr\left(\frac{m}{c_{ii}}\right) = \frac{1}{c_{ii}}Tr(m).$$

Noting that c_{ii} is totally positive, we then see that for any ℓ with $Tr(\ell) \leq Tr(m)$:

$$Tr\left(\frac{\ell}{c_{ii}}\right) = \frac{1}{c_{ii}}Tr(\ell) \leq \frac{1}{c_{ii}}Tr(m) = Tr\left(\frac{m}{c_{ii}}\right).$$

Thus in these cases $N_{(m,Q)} = \left\{\ell \in \mathcal{O}_K^+ | Tr(\ell) \leq Tr(m)\right\}$.

2.5 Increasing Efficiency

A significant piece of code was written to increase the speed of the algorithm. This code essentially discards $\{i_k, j_k\}$ values that are "too large". More specifically,

$$c_{11}y_1^2 + c_{22}y_2^2 + c_{33}y_3^2 + c_{44}y_4^2$$

is a positive definite quadratic form, which means each $c_{ii}y_i^2 \geq 0$. An additional consequence of positive-definiteness is each $\overline{c_{ii}y_i}^2 \geq 0$. Using this:

- Upon determining the ranges of i_4, j_4, discard those which satisfy

$$c_{44}y_4^2 + \overline{c_{44}y_4}^2 > Tr(m).$$

  Call the remaining set A_4. That is,

$$A_4 := \{x_4 = y_4 \leftrightarrow (i_4, j_4) | c_{44}y_4^2 + \overline{c_{44}y_4}^2 \leq Tr(m)\}.$$

- Determine the ranges of i_3, j_3 as described in the previous section but now with $(i_4, j_4) \in A_4$. Consider the set of tuples (i_3, j_3, i_4, j_4). From that set, discard those which satisfy:

$$\begin{aligned} c_{33}y_3^2 + \overline{c_{33}y_3}^2 + c_{44}y_4^2 + \overline{c_{44}y_4}^2 &= Tr\left(c_{33}(i_3 + j_3\theta + a_{34}(i_4 + j_4\theta))^2\right) + Tr\left(c_{44}(i_4 + j_4\theta)^2\right) \\ &> Tr(m). \end{aligned}$$

  Call the remaining set A_3.

This process repeats until the creation of A_1. At this point, the code enumerates over all 8-tuples of A_1, evaluating Q at each tuple.

2.6 The Algorithm

The following algorithm is used to compute $r_Q(\ell)$ for all $\ell \in N_{(m,Q)}$, where Q is a quaternary totally-positive definite integral quadratic form over $K = \mathbb{Q}(\sqrt{d})$ and $m \in \mathcal{O}_K^+$. It has been implemented in Sage ([40]), and the code can be found in the appendix of this document.

Algorithm 2.6.1. *1. Diagonalize $Q(\vec{x})$ over K to obtain a similar form $\widetilde{Q}(\vec{y}) = \langle c_{11}, c_{22}, c_{33}, c_{44}\rangle$ and an upper-triangular change-of-basis matrix $A = (a_{ij})_{1 \leq i \leq j \leq 4}$ where $y_k = \sum_{\ell=k}^{4} a_{k\ell}x_\ell$.*

2. Compute and fix an ordering for the elements of $N(m, Q)$ and initialize $L = [0, ..., 0]$ with $len(L) = |N(m, Q)|$.

3. Recursively generate A_1.

4. *For each* $\vec{w} \in A_1$, *if* $\widetilde{Q}(\vec{w}) = \ell$ *for* $\ell \in N(m,Q)$ *with* $N(m,Q).index(\ell) = n$, *set* $L[n] = L[n] + 1$.

5. *Return* L.

2.7 Examples

In addition to examples in the remainder of this document, we provide the following Sage sample outputs.

Example 2.7.1.

```
x = var('x')

K.<a>=NumberField(x^2-22,embedding=1)

R=Matrix(K,4,[1,1,1,0,  0,1,1,1,  0,0,1,1,  0,0,0,3])


List_for_all_Theta(22,R,5)

[(1, 0), (2, 0), (3, 0), (4, 0), (5, -1), (5, 0), (5, 1)]


Theta_Computation(22,R,5)

[12, 6, 36, 28, 0, 72, 0]
```

The above code considers the number field $K = \mathbb{Q}(\sqrt{22})$. The chosen quadratic form is

$$R(\vec{x}) = x_1^2 + x_1x_2 + x_1x_3 + x_2^2 + x_2x_3 + x_2x_4 + x_3^2 + x_3x_4 + 3x_4^2.$$

In List_for_all_Theta(22,R,5) we take $5 = m \in \mathcal{O}_K^+$ and obtain $N(R,m)$. The output is given in $(a,b) \leftrightarrow a + b\theta$ format. With this particular example, this means that the bounded region for $r_R(5)$ contains the bounded regions of

$$\begin{array}{llllll} (1,0) & \leftrightarrow & 1 & (2,0) & \leftrightarrow & 2 \\ (3,0) & \leftrightarrow & 3 & (4,0) & \leftrightarrow & 4 \\ (5,-1) & \leftrightarrow & 5-\sqrt{22} & (5,0) & \leftrightarrow & 5 \\ (5,1) & \leftrightarrow & 5+\sqrt{22}. & & & \end{array}$$

The command Theta_Computation(22,R,m) returns $r_R(\ell)$ for all $\ell \in N(5, R)$. That is,

$$\begin{array}{rclrcl} r_R(1) & = & 12 & r_R(2) & = & 6 \\ r_R(3) & = & 36 & r_R(4) & = & 28 \\ r_R(5-\sqrt{22}) & = & 0 & r_R(5) & = & 72 \\ r_R(5+\sqrt{22}) & = & 0. & & & \end{array}$$

Example 2.7.2.

```
x = var('x')

K.<a>=NumberField(x^2-101,embedding=1)

R=Matrix(K,4,[1,1,1,0,  0,2,2,-1,  0,0,3,2,  0,0,0,13])


List_for_all_Theta(101,R,15)

[(1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (5, 1), (6, -1), (6, 0), (6,

1), (7, -1), (7, 0), (7, 1), (8, -1), (8, 0), (8, 1), (9, -1), (9, 0),

(9, 1), (10, -1), (10, 0), (10, 1), (10, 2), (11, -1), (11, 0), (11, 1),

(11, 2), (12, -2), (12, -1), (12, 0), (12, 1), (12, 2), (13, -2), (13,

-1), (13, 0), (13, 1), (13, 2), (14, -2), (14, -1), (14, 0), (14, 1),

(14, 2), (15, -2), (15, -1), (15, 0), (16, -2)]


Theta_Computation(101,R,15)

[2, 4, 6, 10, 4, 0, 0, 2, 0, 0, 14, 0, 0, 8, 0, 0, 10, 0, 0, 8, 0, 0, 0,

6, 0, 0, 0, 0, 14, 0, 0, 0, 0, 16, 0, 0, 0, 0, 22, 0, 0, 0, 0, 20, 0]
```

We end with the following two examples, which will be relevant for future chapters of this document:

Example 2.7.3.

```
sage: x = var('x')

sage: K.<a>=NumberField(x^2-5,embedding=1)

sage: S = Matrix(K,4,[1,0,0,0,  0,1,0,0,  0,0,1,0,  0,0,0,1])

sage: List_for_all_Theta(5,S,5)

[(1, 0), (1, 1), (2, -1), (2, 0), (2, 1), (2, 2), (2, 3), (3, -1), (3, 0),

(3, 1), (3, 2), (3, 3), (3, 4), (4, -2), (4, -1), (4, 0),
```

```
(4, 1), (4, 2), (5, -3), (5, -2), (5, -1), (5, 0), (6, -3), (6, -2), (7, -4)]
sage: Theta_Computation(5,S,5)
[8, 8, 8, 24, 48, 24, 8, 48, 80,
96, 96, 80, 48, 24, 96, 216,
160, 144, 8, 96, 160, 248, 80, 144, 48]
```

Example 2.7.4.
```
sage: x = var('x')
sage: K.<a>=NumberField(x^2-5,embedding=1)
sage: T = Matrix(K,4,[1,0,0,0,  0,1,0,0,  0,0,3,0,  0,0,0, 3*(3+a)])
sage: List_for_all_Theta(5,T,5)
[(1, 0), (1, 1), (2, -1), (2, 0), (2, 1), (2, 2), (2, 3), (3, -1), (3, 0),
(3, 1), (3, 2), (3, 3), (3, 4), (4, -2), (4, -1), (4, 0), (4, 1), (4, 2), (5, 0)]
sage: Theta_Computation(5,T,5)
[4, 4, 4, 4, 8, 4, 4, 8, 10,
0, 0, 10, 8, 4, 0, 12, 8, 8, 20]
```

Chapter 3

Quadratic Forms over $\mathbb{Z}$

3.1 Summary

In this chapter, we consider two diagonal forms over $\mathbb{Q}$: $\langle 1,1,1,1\rangle$ and $\langle 1,1,1,7\rangle$. While very well-studied and while their universality follows easily from the $15-$ and $290-$theorems, we provide representation proofs for the following reasons. First, we intend to outline a more general method of proof which will be used in the following chapter regarding forms over $\mathbb{Q}(\sqrt{5})$:

(1) Determine which values $m \in \mathbb{N}$ are locally represented by the quadratic form.

(2) Determine explicitly the Fourier coefficients $\{a_E(m)\}$of the Eisenstein component of the theta-series associated to the quadratic form using local densities. This in turn will involve treating separately:

   (i) The local density at the infinite place.

   (ii) The local densities over any finite places ν_p where p divides neither the level of the form nor m. In turn this involves computing special values of L-functions over $\mathbb{Q}$.

   (iii) The local densities over any finite places ν_p where p divides either the level or m.

(3) Determine the cusp form in the decomposition of the theta series. If the cusp form is not identically zero, determine a bound B such that for all locally represented $m > B$, m is globally represented.

More importantly than outlining a particular method, however, is this: here we provide for the first time the specific Eisenstein series and cusp form comprising the theta series of $\langle 1,1,1,7\rangle$. The proof of the universality of $\langle 1,1,1,7\rangle$ then follows from this decomposition and use of ternary subforms.

3.2 $\langle 1,1,1,1\rangle$

Theorem 3.2.1 (Jacobi, 1834)**.** *Let* $Q(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2$*. For* $m \in \mathbb{N}$*,*

$$r_Q(m) = 8 \cdot \sum_{\substack{0 < d | m \\ 4 \nmid d}} d.$$

We will derive this formula using the theory of local densities.

Introduction

We note that the discriminant of the form is $D_Q = D = 1$, and the level is $N_Q = N = 4$. The character defined by

$$\chi(p) = \left(\frac{D_Q}{p}\right) = \left(\frac{1}{p}\right) = 1$$

is trivial. Since Q has class number one, $r_Q(m) \equiv a_E(m)$ for all m. [38, Zweites Capitel, 11]

The Eisenstein Coefficients

Throughout, by slight abuse of notation, we use p to denote a nonzero prime ideal, its positive generator, and the place ν_p.

We have

$$\begin{aligned} r_Q(m) &= a_E(m) \\ &= \prod_v \beta_v(m) \\ &= \beta_\infty(m)\beta_2(m)\left(\prod_{p|m, p\nmid 2} \beta_p(m)\right)\left(\prod_{p\nmid 2m} \beta_p(m)\right). \end{aligned}$$

We now proceed by separately computing:

- $\beta_\infty(m)$
- $\prod_{p\nmid 2m} \beta_p(m)$ (i.e., $\beta_p(m)$ for all p dividing neither the level nor m)
- $\beta_2(m)$ (i.e., $\beta_p(m)$ for all primes p dividing the level)
- $\prod_{p|m, p\nmid 2} \beta_p(m)$ (all remaining primes)

Lemma 3.2.1. $\beta_\infty(m) = \pi^2 m$.

Proof. This is a direct application of [38, Hilfssatz 72]. □

Lemma 3.2.2. *For each prime $p \nmid 2m$, $\beta_p(m) = 1 - \frac{1}{p^2}$.*

Proof. For odd primes $p \nmid 2m$, the solutions are all of Good type and we have

$$\begin{aligned} \beta_p(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Good}}(m)}{p^{3v}} \\ &= \lim_{v\to\infty} \frac{r_{p^{1+(v-1)}}^{\mathrm{Good}}(m)}{p^{3v}} \\ &= \lim_{v\to\infty} \frac{p^{3(v-1)} r_p^{\mathrm{Good}}(m)}{p^{3v}} \\ &= \frac{r_p^{\mathrm{Good}}(m)}{p^3} \\ &= \frac{p^3 - p}{p^3} = 1 - \frac{1}{p^2}. \end{aligned}$$

□

Thus,

$$\begin{aligned} r_Q(m) &= \beta_\infty(m)\beta_2(m)\left(\prod_{p|m,p\nmid 2}\beta_p(m)\right)\left(\prod_{p\nmid 2m}\beta_p(m)\right) \\ &= \frac{4\pi^2 m}{3}\beta_2(m)\left(\prod_{p|m,p\nmid 2}\frac{\beta_p(m)p^2}{p^2-1}\right)\left(\prod_p 1-\frac{1}{p^2}\right) \\ &= \frac{4\pi^2 m}{3\zeta(2)}\beta_2(m)\left(\prod_{p|m,p\nmid 2}\frac{\beta_p(m)p^2}{p^2-1}\right) \\ &= 8m\beta_2(m)\left(\prod_{p|m,p\nmid 2}\frac{\beta_p(m)p^2}{p^2-1}\right). \end{aligned}$$

Lemma 3.2.3. *For $m \in \mathbb{N}$,*

$$\beta_2(m) = \begin{cases} 1, & 2 \nmid m \\ \dfrac{3}{2^{2n}}, & \mathrm{ord}_2(m) = 2n, n \in \mathbb{N} \\ \dfrac{3}{2^{2n+1}}, & \mathrm{ord}_2(m) = 2n+1, n \geq 0. \end{cases}$$

Proof. When $2 \nmid m$, all solutions are of Good type. Hence

$$\begin{aligned} \beta_2(m) &= \lim_{v\to\infty}\frac{r_{2^v}^{\mathrm{Good}}(m)}{2^{3v}} \\ &= \lim_{v\to\infty}\frac{r_{2^{3+(v-3)}}^{\mathrm{Good}}(m)}{2^{3v}} \\ &= \lim_{v\to\infty}\frac{2^{3(v-3)}r_8^{\mathrm{Good}}(m)}{2^{3v}} \\ &= \frac{512}{2^9} \\ &= 1 \end{aligned}$$

as claimed.

When $\mathrm{ord}_2(m) = 1$, all solutions are still of Good type, and

$$\begin{aligned}
\beta_2(m) &= \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Good}}(m)}{2^{3v}} \\
&= \lim_{v\to\infty} \frac{r_{2^{3+(v-3)}}^{\mathrm{Good}}(m)}{2^{3v}} \\
&= \lim_{v\to\infty} \frac{2^{3(v-3)} r_8^{\mathrm{Good}}(m)}{2^{3v}} \\
&= \frac{768}{2^9} \\
&= \frac{3}{2}.
\end{aligned}$$

When $\mathrm{ord}_2(m) = 2$, we have both Zero and Good type solutions:

$$\begin{aligned}
\beta_2(m) &= \lim_{v\to\infty} \frac{r_{2^v}(m)}{2^{3v}} \\
&= \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Good}}(m)}{2^{3v}} + \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Zero}}(m)}{2^{3v}} \\
&= \frac{1}{2} + \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Zero}}(m)}{2^{3v}} \\
&= \frac{1}{2} + 2^4 \lim_{v\to\infty} \frac{r_{2^{v-2}}^{\mathrm{Good}}(m)}{2^{3v}} \\
&= \frac{1}{2} + \frac{1}{4} = \frac{3}{2^2}.
\end{aligned}$$

For $\mathrm{ord}_2(m) \geq 3$, all solutions are of Zero type. In general when $\mathrm{ord}_2(m) = 2N + 1$ for $N \in \mathbb{N}$:

$$\begin{aligned}
\beta_2(m) &= \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Zero}}(m)}{2^{3v}} \\
&= \lim_{v\to\infty} \frac{2^{4N} r_{2^{v-2N}}^{\mathrm{Good}}(2)}{2^{3v}} \\
&= \frac{1}{2^{2N}} \cdot \frac{3}{2} \\
&= \frac{3}{2^{2N+1}}.
\end{aligned}$$

Similarly, when $\mathrm{ord}_2(m) = 2N+2$ for $N \in \mathbb{N}$

$$\begin{aligned}
\beta_2(m) &= \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Zero}}(m)}{2^{3v}} \\
&= \lim_{v\to\infty} \frac{2^{4N} r_{2^{v-2N}}(4)}{2^{3v}} \\
&= \frac{1}{2^{2N}} \lim_{v\to\infty} \frac{r_{2^{v-2N}}(4)}{2^{3(v-2N)}} \\
&= \frac{1}{2^{2N}} \cdot \frac{3}{4} \\
&= \frac{3}{2^{2N+2}}.
\end{aligned}$$

This completes the proof of the claim. □

Lemma 3.2.4. *For odd $p|m$*

$$\frac{\beta_p(m)p^2}{p^2-1} = \sum_{i=0}^{\mathrm{ord}_p(m)} p^{-i}.$$

Proof. Suppose $\mathrm{ord}_p(m) = 1$. Then all solutions are of Good type and

$$\begin{aligned}
\beta_p(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Good}}(m)}{p^{3v}} \\
&= \lim_{v\to\infty} \frac{p^{3(v-1)} r_p^{\mathrm{Good}}(m)}{p^{3v}} \\
&= \frac{r_p^{\mathrm{Good}}(m)}{p^3} \\
&= \frac{p^3 + p(p-1) - 1}{p^3}.
\end{aligned}$$

Thus, for such primes we have

$$\begin{aligned}
\frac{\beta_p(m)p^2}{p^2-1} &= \frac{p^3 + p(p-1) - 1}{p\,(p^2-1)} \\
&= 1 + \frac{1}{p} = \sum_{i=0}^{\mathrm{ord}_p(m)} p^{-i}.
\end{aligned}$$

In the case that $\mathrm{ord}_p(m) = 2N$ ($N \in \mathbb{N}$), both Good and Zero type solutions exist with

$$\begin{aligned}
\beta_p(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Good}}(m)}{p^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Zero}}(m)}{p^{3v}} \\
&= \frac{p^3+p(p-1)-1}{p^3} + \lim_{v\to\infty} \frac{1}{p^{3v}} \left(\left(\sum_{i=1}^{N-1} p^{4i} r_{p^{v-2i}}^{\mathrm{Good}}(m/p^{2i}) \right) + p^{4N} r_{p^{v-2N}}^{\mathrm{Good}}(m/p^{2N}) \right) \\
&= \left(\sum_{i=0}^{N-1} p^{-2i} \right) \left(\frac{p^3+p(p-1)-1}{p^3} \right) + p^{-2N} \left(1 - 1/p^2\right).
\end{aligned}$$

So for each such p:

$$\begin{aligned}
\frac{\beta_p(m)p^2}{p^2-1} &= \left(\sum_{i=0}^{N-1} p^{-2i} \right) \left(1 + \frac{1}{p} \right) + p^{-2N} \\
&= \sum_{i=0}^{2N} p^{-i} = \sum_{i=0}^{\mathrm{ord}_p(m)} p^{-i}.
\end{aligned}$$

Last, suppose $\mathrm{ord}_p(m) = 2N+1$ (where $N \in \mathbb{N}$). Again only Good and Zero type solutions exist with

$$\begin{aligned}
\beta_p(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Good}}(m)}{p^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Zero}}(m)}{p^{3v}} \\
&= \frac{p^3+p(p-1)-1}{p^3} + \lim_{v\to\infty} \frac{1}{p^{3v}} \left(\sum_{i=1}^{N} p^{4i} r_{p^{v-2i}}^{\mathrm{Good}}(m/p^{2i}) \right) \\
&= \left(\sum_{i=0}^{N} p^{-2i} \right) \left(\frac{p^3+p(p-1)-1}{p^3} \right).
\end{aligned}$$

Hence for such p:

$$\begin{aligned}
\frac{\beta_p(m)p^2}{p^2-1} &= \left(\sum_{i=0}^{N} p^{-2i} \right) \left(1 + \frac{1}{p} \right) \\
&= \sum_{i=0}^{2N+1} p^{-i} = \sum_{i=0}^{\mathrm{ord}_p(m)} p^{-i}.
\end{aligned}$$

□

Proof of Theorem

Suppose $m \in \mathbb{N}$ is odd. Then

$$\begin{aligned} r_Q(m) &= 8m\beta_2(m)\left(\prod_{p|m,p\nmid 2}\frac{\beta_p(m)p^2}{p^2-1}\right) \\ &= 8m\prod_{p|m}\left(\sum_{i=0}^{\mathrm{ord}_p(m)}p^{-i}\right) \\ &= 8\prod_{p|m}\left(\sum_{i=0}^{\mathrm{ord}_p(m)}p^{i}\right) \\ &= 8\cdot \sum_{\substack{0<d|m \\ 4\nmid d}} d. \end{aligned}$$

Similarly, for $m \in \mathbb{N}$ even

$$\begin{aligned} r_Q(m) &= 8m\beta_2(m)\left(\prod_{p|m,p\nmid 2}\frac{\beta_p(m)p^2}{p^2-1}\right) \\ &= 8m\left(\frac{3}{2^{\mathrm{ord}_2(m)}}\right)\prod_{2\neq p|m}\left(\sum_{i=0}^{\mathrm{ord}_p(m)}p^{-i}\right) \\ &= 8(2+1)\prod_{2\neq p|m}\left(\sum_{i=0}^{\mathrm{ord}_p(m)}p^{i}\right) \\ &= 8\cdot \sum_{\substack{0<d|m \\ 4\nmid d}} d \end{aligned}$$

as claimed.

3.3 $\langle 1,1,1,7\rangle$

Theorem 3.3.1. *The form $Q(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + 7x_4^2$ is positive universal over $\mathbb{Z}$. That is, for all $m \in \mathbb{N}$, $Q(\vec{x}) = m$ has a solution with $\vec{x} \in \mathbb{Z}^4$.*

Preliminaries

For this particular form, the discriminant is $D_Q = D = 7$ and the norm is $N_Q = N = 28$. In addition, the associated character is

$$\begin{aligned}\chi(p) &= \left(\frac{(-1)^2\det(Q)}{p}\right)\\ &= \left(\frac{7}{p}\right)\\ &= \begin{cases}1, & p \equiv 1,3,9,19,25,27 \pmod{28}\\ -1, & p \equiv 5,11,13,15,17,23 \pmod{28}.\end{cases}\end{aligned}$$

Note that as this is a character of modulus 28, we have $\chi(2) = \chi(7) = 0$.

The Eisenstein Coefficients

Recall by Siegel's product formula [38, pg. 285] we have

$$\begin{aligned}a_E(m) &= \prod_v \beta_v(m)\\ &= \beta_\infty(m)\prod_{p<\infty}\beta_p(m)\\ &= \beta_\infty(m)\beta_2(m)\beta_7(m)\left(\prod_{2,7\neq p\mid m}\beta_p(m)\right)\left(\prod_{2,7\neq q\nmid m}\beta_q(m)\right).\end{aligned}$$

Again, in order to determine explicitly $a_E(m)$ for all $m \in \mathbb{N}$ we proceed by computing separately:

- $\beta_\infty(m)$
- $\prod_{2,7=q\nmid m}\beta_q(m)$ (the local densities at primes dividing neither the level nor m)

- $\beta_2(m)$ and $\beta_7(m)$ (i.e., $\beta_p(m)$ for all primes p which divide the level)
- $\prod_{2,7\neq p|m} \beta_p(m)$ (the local densities at all remaining primes).

Lemma 3.3.1. *For all $m \in \mathbb{N}$, $\beta_\infty(m) = \frac{\pi^2 m}{\sqrt{7}}$.*

Proof. This is a direct application of [38, Hilfssatz 72]. □

Lemma 3.3.2. *For finite primes $q \nmid 14m$*

$$\beta_q(m) = \left(1 - \frac{\chi(q)}{q^2}\right).$$

Proof. Because all solutions are of Good type

$$\begin{aligned}
\beta_q(m) &= \lim_{v\to\infty} \frac{r_{q^v}^{\mathrm{Good}}(m)}{q^{3v}} \\
&= \lim_{v\to\infty} \frac{q^{3(v-1)} r_q^{\mathrm{Good}}(m)}{q^{3v}} \\
&= \frac{q^3 - \chi(q)q}{q^3} \\
&= 1 - \frac{\chi(q)}{q^2}.
\end{aligned}$$

□

Hence

$$\begin{aligned}
a_E(m) &= \beta_\infty(m)\beta_2(m)\beta_7(m)\left(\prod_{2,7\neq p|m}\beta_p(m)\right)\left(\prod_{2,7\neq q\nmid m}\beta_q(m)\right)\\
&= \beta_\infty(m)\beta_2(m)\beta_7(m)\left(\prod_{2,7\neq p|m}\beta_p(m)\right)\prod_{q\nmid 14m}\left(1-\chi(q)/q^2\right)\\
&= \frac{\pi^2 m}{\sqrt{7}}\beta_2(m)\beta_7(m)\left(\prod_{2,7\neq p|m}\beta_p(m)\right)\frac{\prod_q\left(1-\chi(q)/q^2\right)}{\prod_{p|14m}\left(1-\chi(p)/p^2\right)}\\
&= \frac{\pi^2 m}{\sqrt{7}}(L_{\mathbb{Q}}(2,\chi))^{-1}\left(\frac{\beta_2(m)}{1-\chi(2)/2^2}\right)\left(\frac{\beta_7(m)}{1-\chi(7)/7^2}\right)\left(\prod_{2,7\neq p|m}\frac{\beta_p(m)p^2}{p^2-\chi(p)}\right)\\
&= \frac{\pi^2 m}{\sqrt{7}}(L_{\mathbb{Q}}(2,\chi))^{-1}\beta_2(m)\beta_7(m)\left(\prod_{2,7\neq p|m}\frac{\beta_p(m)p^2}{p^2-\chi(p)}\right).
\end{aligned}$$

Lemma 3.3.3. $L_{\mathbb{Q}}(2,\chi)=\dfrac{2\sqrt{7}\pi^2}{49}$.

Proof. Using the notation of Iwasawa [23, pg. 104], we have:

$$L_{\mathbb{Q}}(2,\chi)=-\frac{2\pi^2}{(28)^2}\tau(\chi)L_{\mathbb{Q}}(1-2,\overline{\chi})$$

where

$$\begin{aligned}
L_{\mathbb{Q}}(1-2,\overline{\chi}) &= L_{\mathbb{Q}}(1-2,\chi)\\
&= -\frac{28}{2}\left(\sum_{a=1}^{28}\chi(a)\left(\left(\frac{a-28}{28}\right)^2+\left(\frac{a-28}{28}\right)+\frac{1}{6}\right)\right)\\
&= -\frac{28}{2}\left(\frac{448}{(28)^2}\right)=-8,
\end{aligned}$$

and where

$$\begin{aligned}
\tau(\chi) &= \sum_{a=1}^{28}\chi(a)e^{2\pi ia/28}\\
&= 2\sqrt{7}.
\end{aligned}$$

Hence

$$L(2,\chi) = \frac{2\sqrt{7}\pi^2}{49}$$

as originally claimed. □

Lemma 3.3.4.

$$\beta_2(m) = \begin{cases} \frac{3}{4}\left(\sum_{i=0}^{N}\frac{1}{2^{2i}}\right) + \begin{cases} 1/2^{2N+1}, & m/2^{2N} \equiv 2 \pmod 8 \\ 0, & m/2^{2N} \equiv 6 \pmod 8 \end{cases} \\ \frac{3}{4}\left(\sum_{i=0}^{N-1}\frac{1}{2^{2i}}\right) + \begin{cases} 3/2^{2N+1}, & m/2^{2N} \equiv 1 \pmod 4 \\ 1/2^{2N+1}, & m/2^{2N} \equiv 3 \pmod 4. \end{cases} \end{cases}$$

Proof. If m is odd, then all solutions are of Good type. Thus

$$\begin{aligned} \beta_2(m) &= \lim_{v\to\infty}\frac{r_{2^v}^{\text{Good}}(m)}{2^{3v}} \\ &= \lim_{v\to\infty}\frac{2^{3(v-6)}r_{2^6}^{\text{Good}}(m)}{2^{3v}} \\ &= \frac{r_{2^6}^{\text{Good}}(m)}{2^{18}} \\ &= \begin{cases} 3/2, & m \equiv 1 \pmod 4 \\ 1/2, & m \equiv 3 \pmod 4. \end{cases} \end{aligned}$$

When $\text{ord}_2(m) = 1$, all solutions again are of Good type and we similarly have

$$\begin{aligned} \beta_2(m) &= \frac{r_{2^6}^{\text{Good}}(m)}{2^{18}} \\ &= \begin{cases} 5/4, & m \equiv 2 \pmod 8 \\ 3/4, & m \equiv 6 \pmod 8. \end{cases} \end{aligned}$$

Suppose $\mathrm{ord}_2(m) = 2$. This implies $m \equiv 4 \pmod 8$ or $m/4 \equiv 1, 3 \pmod 4$. Here both Good and Zero type solutions occur and

$$\begin{aligned}
\beta_2(m) &= \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Good}}(m)}{2^{3v}} + \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Zero}}(m)}{2^{3v}} \\
&= \frac{r_{2^6}^{\mathrm{Good}}(m)}{2^{18}} + \lim_{v\to\infty} \frac{2^4 r_{2^{v-2}}(m/4)}{2^{3v}} \\
&= \frac{r_{2^6}^{\mathrm{Good}}(m)}{2^{18}} + \lim_{v\to\infty} \frac{2^4 r_{2^{v-2}}^{\mathrm{Good}}(m/4)}{2^{3v}} \\
&= \frac{r_{2^6}^{\mathrm{Good}}(m)}{2^{18}} + \frac{r_{2^6}^{\mathrm{Good}}(m/4)}{2^{20}} \\
&= \begin{cases} 3/4 + 3/8, & m \equiv 4, 20, 36, 52 \pmod{64} \\ 3/4 + 1/8, & m \equiv 12, 28, 44, 60 \pmod{64} \end{cases} \\
&= \begin{cases} 9/8, & m \equiv 4, 20, 36, 52 \pmod{64} \\ 7/8, & m \equiv 12, 28, 44, 60 \pmod{64} \end{cases} \\
&= \begin{cases} 9/8, & m/4 \equiv 1 \pmod 4 \\ 7/8, & m/4 \equiv 3 \pmod 4. \end{cases}
\end{aligned}$$

Next suppose $\mathrm{ord}_2(m) = 2N+1$, $N \in \mathbb{Z}^{\geq 0}$. This means $\mathrm{ord}_2(m/2N) = 1$ or $m/2N \equiv 2, 6 \pmod 8$. As there are only Zero and Good type solutions:

$$\begin{aligned}
\beta_2(m) &= \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Good}}(m)}{2^{3v}} + \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Zero}}(m)}{2^{3v}} \\
&= \frac{3}{4} + \lim_{v\to\infty} \frac{1}{2^{3v}} \cdot \left(\sum_{i=1}^{N} 2^{4i} 2^{3(v-2i-6)} r_{2^6}^{\mathrm{Good}}(m/2^{2i}) \right) \\
&= \frac{3}{4} + \left(\sum_{i=1}^{N-1} \frac{r_{2^6}^{\mathrm{Good}}(m/2^{2i})}{2^{2i}2^{18}} \right) + \frac{r_{2^6}^{\mathrm{Good}}(m/2^{2N})}{2^{2N}2^{18}} \\
&= \frac{3}{4} \left(\sum_{i=0}^{N-1} \frac{1}{2^{2i}} \right) + \frac{1}{2^{2N}} \begin{cases} 5/4, & m/2^{2N} \equiv 2 \pmod 8 \\ 3/4, & m/2^{2N} \equiv 6 \pmod 8 \end{cases} \\
&= \frac{3}{4} \left(\sum_{i=0}^{N} \frac{1}{2^{2i}} \right) + \begin{cases} 1/2^{2N+1}, & m/2^{2N} \equiv 2 \pmod 8 \\ 0, & m/2^{2N} \equiv 6 \pmod 8. \end{cases}
\end{aligned}$$

Finally if $\mathrm{ord}_2(m) = 2N$, $N \in \mathbb{Z}^{\geq 0}$, only Good and Zero type solutions are present with

$$\begin{aligned}
\beta_2(m) &= \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Good}}(m)}{2^{3v}} + \lim_{v\to\infty} \frac{r_{2^v}^{\mathrm{Zero}}(m)}{2^{3v}} \\
&= \frac{3}{4} + \lim_{v\to\infty} \frac{1}{2^{3v}} \cdot \left(\sum_{i=1}^{N} 2^{4i} 2^{3(v-2i-6)} r_{2^6}^{\mathrm{Good}}(m/2^{2i}) \right) \\
&= \frac{3}{4} \left(\sum_{i=0}^{N-1} \frac{1}{2^{2i}} \right) + \frac{1}{2^{2N}} \begin{cases} 3/2, & m/2^{2N} \equiv 1 \pmod 4 \\ 1/2, & m/2^{2N} \equiv 3 \pmod 4 \end{cases} \\
&= \frac{3}{4} \left(\sum_{i=0}^{N-1} \frac{1}{2^{2i}} \right) + \begin{cases} 3/2^{2N+1}, & m/2^{2N} \equiv 1 \pmod 4 \\ 1/2^{2N+1}, & m/2^{2N} \equiv 3 \pmod 4. \end{cases}
\end{aligned}$$

□

Lemma 3.3.5.

$$\beta_7(m) = \begin{cases} 50\left(\dfrac{7^{2N}-1}{7^{2(N+1)}}\right) + \begin{cases} 6/7^{2N+1}, & m/7^{2N} \equiv 1,2,4 \pmod 7 \\ 8/7^{2N+1}, & m/7^{2N} \equiv 3,5,6 \pmod 7 \end{cases} \\ 50\left(\dfrac{7^{2N}-1}{7^{2(N+1)}}\right) + \begin{cases} 342/7^{2N+3}, & m/7^{2N+1} \equiv 1,2,4 \pmod 7 \\ 2392/7^{2N+4}, & m/7^{2N+1} \equiv 3,5,6 \pmod 7. \end{cases} \end{cases}$$

Proof. If $m \not\equiv 0 \pmod 7$, all solutions are of Good type and

$$\begin{aligned} \beta_7(m) &= \lim_{v\to\infty} \frac{r_{7^v}^{\text{Good}}(m)}{7^{3v}} \\ &= \lim_{v\to\infty} \frac{7^{3(v-1)} r_7^{\text{Good}}(m)}{7^{3v}} \\ &= \frac{r_7^{\text{Good}}(m)}{7^3} \\ &= \begin{cases} 6/7, & m \equiv 1,2,4 \pmod 7 \\ 8/7, & m \equiv 3,5,6 \pmod 7. \end{cases} \end{aligned}$$

When $\mathrm{ord}_7(m) = 1$, we have the potential for both Good and Bad-Type-I solutions. Consequently,

$$\begin{aligned}
\beta_7(m) &= \lim_{v\to\infty} \frac{r_{7^v}^{\text{Good}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{r_{7^v}^{\text{Bad-Type-I}}(m)}{7^{3v}} \\
&= \lim_{v\to\infty} \frac{7^{3(v-1)} r_7^{\text{Good}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{7 r_{7^{v-1},Q'}^{\text{Good}}(m/7)}{7^{3v}} \\
&= \frac{r_7^{\text{Good}}(m)}{7^3} + \lim_{v\to\infty} \frac{7\cdot 7^{3(v-2)} r_{7,Q'}^{\text{Good}}(m/7)}{7^{3v}} \\
&= \frac{336}{343} + \frac{r_{7,Q'}^{\text{Good}}(m/7)}{7^5} \\
&= \frac{336}{343} + \begin{cases} 6/7^3, & m/7 \equiv 1,2,4 \pmod 7 \\ 40/7^4, & m/7 \equiv 3,5,6 \pmod 7 \end{cases} \\
&= \begin{cases} 342/7^3, & m/7 \equiv 1,2,4 \pmod 7 \\ 2392/7^4, & m/7 \equiv 3,5,6 \pmod 7 \end{cases}
\end{aligned}$$

where $Q' = \langle 1,1,1,1\rangle$.

When $\mathrm{ord}_7(m) = 2$, we have Good, Bad-Type-I, and Zero-solutions with

$$\begin{aligned}
\beta_7(m) &= \lim_{v\to\infty} \frac{r_{7^v}^{\text{Good}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{r_{7^v}^{\text{Bad-Type-I}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{r_{7^v}^{\text{Zero}}(m)}{7^{3v}} \\
&= \frac{336}{343} + \lim_{v\to\infty} \frac{7 r_{7^{v-1},Q'}^{\text{Good}}(m/7)}{7^{3v}} + \lim_{v\to\infty} \frac{7^4 r_{7^{v-2}}(m/7^2)}{7^{3v}} \\
&= \frac{336}{343} + \lim_{v\to\infty} \frac{7\cdot 7^{3(v-2)} r_{7,Q'}^{\text{Good}}(m/7)}{7^{3v}} + \lim_{v\to\infty} \frac{7^4 7^{3(v-3)} r_{7}^{\text{Good}}(m/7^2)}{7^{3v}} \\
&= \frac{336}{343} + \frac{336}{7^5} + \frac{r_{7}^{\text{Good}}(m/7^2)}{7^5} \\
&= \frac{336}{343} + \frac{336}{7^5} + \begin{cases} 294/7^5, & m/7^2 \equiv 1,2,4 \pmod 7 \\ 392/7^5, & m/7^2 \equiv 3,5,6 \pmod 7 \end{cases} \\
&= \begin{cases} 2442/2401, & m/7^2 \equiv 1,2,4 \pmod 7 \\ 2456/2401, & m/7^2 \equiv 3,5,6 \pmod 7 \end{cases} \\
&= 50\left(\frac{7^2-1}{7^4}\right) + \begin{cases} 6/7^3, & m/7^2 \equiv 1,2,4 \pmod 7 \\ 8/7^3, & m/7^2 \equiv 3,5,6 \pmod 7. \end{cases}
\end{aligned}$$

Suppose now that $\mathrm{ord}_7(m) = 2N$, $N \in \mathbb{N} \cup \{0\}$. In this case:

$$\begin{aligned}
\beta_7(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\text{Good}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\text{Bad-Type-I}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\text{Zero}}(m)}{7^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{7^{3v}} \cdot \left(\sum_{i=0}^{N-1} 7^{4i} r_{7^{v-2i}}^{\text{Good}}(m/7^{2i}) \right) \\
&\quad + \lim_{v\to\infty} \frac{1}{7^{3v}} \cdot \left(\sum_{i=0}^{N-1} 7^{4i} r_{7^{v-2i}}^{\text{Bad-Type-I}}(m/7^{2i}) \right) + \lim_{v\to\infty} \frac{7^{4N} r_{v^{7-2N}}(m/7^{2N})}{7^{3v}} \\
&= \left(\frac{336}{343}\right) \left(\sum_{i=0}^{N-1} \frac{1}{7^{2i}} \right) + \left(\frac{336}{7^5}\right) \left(\sum_{i=0}^{N-1} \frac{1}{7^{2i}} \right) + \frac{r_7^{\text{Good}}(m/7^{2N})}{7^3 7^{2N}} \\
&= \left(\frac{336}{343}\right) \left(\sum_{i=0}^{N-1} \frac{1}{7^{2i}} \right) + \left(\frac{336}{7^5}\right) \left(\sum_{i=0}^{N-1} \frac{1}{7^{2i}} \right) + \frac{1}{7^{2N}} \begin{cases} 6/7, & m/7^{2N} \equiv 1,2,4 \pmod 7 \\ 8/7, & m/7^{2N} \equiv 3,5,6 \pmod 7 \end{cases} \\
&= \left(\frac{50}{2401}\right) \left(\frac{7^{2N}-1}{7^{2(N-1)}} \right) + \frac{1}{7^{2N}} \begin{cases} 6/7, & m/7^{2N} \equiv 1,2,4 \pmod 7 \\ 8/7, & m/7^{2N} \equiv 3,5,6 \pmod 7 \end{cases} \\
&= 50 \left(\frac{7^{2N}-1}{7^{2(N+1)}} \right) + \begin{cases} 6/7^{2N+1}, & m/7^{2N} \equiv 1,2,4 \pmod 7 \\ 8/7^{2N+1}, & m/7^{2N} \equiv 3,5,6 \pmod 7. \end{cases}
\end{aligned}$$

Last, suppose $\text{ord}_7(m) = 2N+1$, $N \in \mathbb{N} \cup \{0\}$. Then

$$\begin{aligned}
\beta_7(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\text{Good}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\text{Bad-Type-I}}(m)}{7^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\text{Zero}}(m)}{7^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{7^{3v}} \cdot \left(\sum_{i=0}^{N-1} 7^{4i} r_{7^{v-2i}}^{\text{Good}}(m/7^{2i}) \right) \\
&\quad + \lim_{v\to\infty} \frac{1}{7^{3v}} \cdot \left(\sum_{i=0}^{N-1} 7^{4i} r_{7^{v-2i}}^{\text{Bad-Type-I}}(m/7^{2i}) \right) + \lim_{v\to\infty} \frac{7^{4N} r_{v^{7-2N}}(m/7^{2N})}{7^{3v}} \\
&= \left(\frac{336}{343}\right) \left(\sum_{i=0}^{N} \frac{1}{7^{2i}} \right) + \left(\frac{336}{7^5}\right) \left(\sum_{i=0}^{N-1} \frac{1}{7^{2i}} \right) + \frac{1}{7^{2N+1}} \begin{cases} 294/7^4, & m/7^{2N+1} \equiv 1,2,4 \pmod 7 \\ 280/7^4, & m/7^{2N+1} \equiv 3,5,6 \pmod 7 \end{cases} \\
&= \left(\frac{50}{2401}\right) \left(\frac{7^{2N}-1}{7^{2(N-1)}} \right) + \frac{1}{7^{2N+1}} \begin{cases} 342/49, & m/7^{2N+1} \equiv 1,2,4 \pmod 7 \\ 2392/343, & m/7^{2N+1} \equiv 3,5,6 \pmod 7 \end{cases} \\
&= 50 \left(\frac{7^{2N}-1}{7^{2(N+1)}} \right) + \begin{cases} 342/7^{2N+3}, & m/7^{2N+1} \equiv 1,2,4 \pmod 7 \\ 2392/7^{2N+4}, & m/7^{2N+1} \equiv 3,5,6 \pmod 7. \end{cases}
\end{aligned}$$

□

Lemma 3.3.6. *Let $N \in \mathbb{N} \cup \{0\}$. Let $p \neq 2,7$ be prime with $p|m$. Then:*

$$\beta_p(m) \cdot \frac{p^2}{p^2 - \chi(p)} = \begin{cases} \frac{1}{p^{2N}} \left(\frac{p^{2N+1}-1}{p-1} \right), & ord_p(m) = 2N, \chi(p) = 1 \\ \frac{1}{p^{2N}} \left(\frac{p^{2N+1}+1}{p+1} \right), & ord_p(m) = 2N, \chi(p) = -1 \\ \frac{1}{p^{2N+1}} \left(\frac{p^{2N+2}-1}{p-1} \right), & ord_p(m) = 2N+1, \chi(p) = 1 \\ \frac{1}{p^{2N+1}} \left(\frac{p^{2N+2}-1}{p+1} \right), & ord_p(m) = 2N+1, \chi(p) = -1. \end{cases}$$

Proof. Suppose $\mathrm{ord}_p(m) = 2N$ ($N \in \mathbb{N}$). Here we have the potential for both Good and Zero type solutions and

$$\begin{aligned}
\beta_p(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Good}}(m)}{p^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Zero}}(m)}{p^{3v}} \\
&= \frac{p^3 + p(p-1)\chi(p) - 1}{p^3} + \lim_{v\to\infty} \frac{1}{p^{3v}} \left(\sum_{i=1}^{N-1} p^{4i} r_{p^{v-2i}}^{\mathrm{Good}}(m/p^{2i}) + p^{4N} r_{p^{v-2N}}(m/p^{2N}) \right) \\
&= \left(\sum_{i=0}^{N-1} p^{-2i} \right) \left(\frac{p^3 + p(p-1)\chi(p) - 1}{p^3} \right) + p^{-2N} \left(1 - \chi(p)/p^2 \right)
\end{aligned}$$

Hence

$$\begin{aligned}
\beta_p(m) \cdot \frac{p^2}{p^2 - \chi(p)} &= \begin{cases} \sum_{i=0}^{2N} p^{-i}, & \chi(p) = 1 \\ \sum_{i=0}^{2N} (-1)^i p^{-i}, & \chi(p) = -1 \end{cases} \\
&= \begin{cases} \frac{1}{p^{2N}} \left(\frac{p^{2N+1} - 1}{p-1} \right), & \chi(p) = 1 \\ \frac{1}{p^{2N}} \left(\frac{p^{2N+1} + 1}{p+1} \right), & \chi(p) = -1. \end{cases}
\end{aligned}$$

Next, suppose $\mathrm{ord}_p(m) = 2N + 1$ ($N \geq 0$). Again, here we have Good and Zero type solutions with

$$\begin{aligned}
\beta_p(m) &= \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Good}}(m)}{p^{3v}} + \lim_{v\to\infty} \frac{r_{p^v}^{\mathrm{Zero}}(m)}{p^{3v}} \\
&= \frac{p^3 + p(p-1)\chi(p) - 1}{p^3} + \lim_{v\to\infty} \frac{1}{p^{3v}} \left(\sum_{j=1}^{N} p^{4i} p^{3(v-2i-1)} (p^3 + p(p-1)\chi(p) - 1) \right) \\
&= \frac{p^3 + p(p-1)\chi(p) - 1}{p^3} + \sum_{i=1}^{N} p^{-2i} \left(\frac{p^3 + p(p-1)\chi(p) - 1}{p^3} \right).
\end{aligned}$$

Thus

$$\begin{aligned}\beta_p(m)\cdot\frac{p^2}{p^2-\chi(p)} &= \begin{cases}\sum_{i=0}^{2N+1} p^{-i}, & \chi(p)=1\\ \sum_{i=0}^{2N+1}(-1)^i p^{-i}, & \chi(p)=-1\end{cases}\\ &= \begin{cases}\frac{1}{p^{2N+1}}\left(\frac{p^{2N+2}-1}{p-1}\right), & \chi(p)=1\\ \frac{1}{p^{2N+1}}\left(\frac{p^{2N+2}-1}{p+1}\right), & \chi(p)=-1.\end{cases}\end{aligned}$$

□

This series of lemmas, then, completely determines $a_E(m)$ for any $m \in \mathbb{N}$.

The Cusp Coefficients

Because we have an explicit formula for $a_E(m)$ for any locally represented m, and because for "small" m the representation numbers $r_Q(m)$ can be computed by hand, we can determine the coefficients of any existing cusp components:

Table 3.1: Table of $r_Q(m)$, $a_E(m)$, $a_C(m)$ for small m

m	$r_Q(m)$	$a_E(m)$	$a_C(m) = r_Q(m) - a_E(m)$
1	6	9/2	3/2
2	12	15/2	9/2
3	8	8	0
4	6	27/2	-15/2
5	24	24	0
6	24	24	0
9	54	117/2	- 9/2
10	40	40	0

We now use Sage [40] to compute a basis for the space of cusps forms of weight 2, level 28 and character χ:

```
sage: G = DirichletGroup(28)
sage: G[7]
Dirichlet character modulo 28 of conductor 28 mapping 15 |--> -1, 17 |--> -1


sage: C=CuspForms(G[7],2)
sage: C.set_precision(100)
sage: C.basis()
[q - 2*q^4 - q^7 + 2*q^8 - 3*q^9 + 2*q^11 + 4*q^14 +
2*q^16 - 8*q^22 - 2*q^23 + 5*q^25 - 2*q^28
 - 2*q^29 - 6*q^32 + 6*q^36 + 6*q^37 + 2*q^43
 + 4*q^44 + 8*q^46 - 7*q^49 - 10*q^53 - 6*q^56 +
3*q^63 + 2*q^64 - 6*q^67 - 2*q^71 - 6*q^72 +
14*q^77 + 6*q^79 + 9*q^81 - 8*q^86 + 12*q^88
- 4*q^92 - 6*q^99 + O(q^100),


q^2 - q^4 - 2*q^7 - q^8 + 4*q^11 + q^14 + 3*q^16 -
3*q^18 - 2*q^22 - 4*q^23 + 3*q^28 - q^32
 + 3*q^36 + 4*q^43 - 6*q^44 + 2*q^46 +
 5*q^50 - 5*q^56 - 2*q^58 + 6*q^63 - 5*q^64 - 12*q^67
 - 4*q^71 + 3*q^72 + 6*q^74 + 12*q^79 -
 2*q^86 + 10*q^88 + 6*q^92 - 7*q^98 - 12*q^99 + O(q^100)]
```

This means that the space of cusp forms is two dimensional, with basis

$$\begin{aligned} f_0(q) &= q - 2q^2 + 0q^3 + 0q^4 + 0q^5 + 0q^6 - q^7 + 2q^8 - 3q^9 + 0q^{10} + 2q^{11} + O(q^{12}) \\ f_1(q) &= 0q + q^2 + 0q^3 - q^4 + 0q^5 + 0q^6 - 2q^7 - q^8 + 0q^9 + +0q^{10} + 4q^{11} + O(q^{12}) \end{aligned}$$

To obtain a Hecke eigenbasis from the given basis, consider the action of T_{11}:

$$\begin{aligned}
T_{11}f_0(\tau) &= 0 + (a_{11}f_0)q + (a_{22}f_0)q^2 + (a_{33}f_0)q^3 + (a_{44}f_0)q^4 + (a_{55}f_0)q^5 + ... \\
&= 0 + 2q - 8q^2 + 0q^3 + 4q^4 + ... \\
&= 2f_0 - 8f_1 \\
T_{11}f_1(\tau) &= 0 + (a_{11}f_1)q + (a_{22}f_1)q^2 + (a_{33}f_1)q^3 + (a_{44}f_1)q^4 + (a_{55}f_1)q^5 + ... \\
&= 0 + 4q - 2q^2 + 0q^3 - 6q^4 + ... \\
&= 4f_0 - 2f_1
\end{aligned}$$

The corresponding matrix

$$M_{11} := \begin{bmatrix} 2 & -8 \\ 4 & -2 \end{bmatrix}$$

has eigenvalues $\lambda = \pm\sqrt{28}i$, with eigenspaces parameterized by $t_1 \begin{bmatrix} 1 \\ \frac{1-\sqrt{7}i}{4} \end{bmatrix}$ and $t_2 \begin{bmatrix} 1 \\ \frac{1+\sqrt{7}i}{4} \end{bmatrix}$.

Setting $\alpha := (1+\sqrt{7}i)/4$, we then have the eigenbasis:

$$\begin{aligned}
F_0 &:= f_0 + \overline{\alpha}f_1 \\
&= q + \overline{\alpha}q^2 + 0q^3 + (-2-\overline{\alpha})q^4 + 0q^5 + 0q^6 + (-1-2\overline{\alpha})q^7 + ... \\
F_1 &:= f_0 + \alpha f_1 \\
&= q + \alpha q^2 + 0q^3 + (-2-\alpha)q^4 + 0q^5 + 0q^6 + (-1-2\alpha)q^7 + (2-\alpha)q^8 + ...
\end{aligned}$$

In particular, we have shown that $\sum a_C(m)q^m = \gamma_0 F_0 + \gamma_1 F_1$ where

$$\begin{aligned}
\gamma_0 &= \frac{21+33\sqrt{7}i}{28} \\
\gamma_1 &= \frac{21-33\sqrt{7}i}{28}.
\end{aligned}$$

Proving Universality

Theorem 6.3 of [21] applied to $\langle 1,1,1,7\rangle$ guarantees the representation of any $m \in \mathbb{N}$ satisfying

$$\sqrt{(m)}/\tau(m) \prod_{p\nmid N, p|m, \chi(p)=-1} \frac{p-1}{p+1} > \frac{12\sqrt{14}}{7\widehat{\Lambda}}$$

where

$$\widehat{\Lambda} \;=\; \frac{3}{2}\min_{T'}\left\{\prod_{p|N} C'_p(T')\right\}.$$

The T' range over the square classes $v \in \mathbb{Z}_p^{\times}/\mathbb{Z}_p^{\times 2}$ for $p|N$. For $\mathbb{Z}_2$, we take coset representatives of $\{\pm 1, \pm 2, \pm 5, \pm 10\}$ and for $\mathbb{Z}_7$, we take $\{1,3,7,21\}$ as our coset representatives.

By definition (c.f., [21, pg. 369])

$$C'_p(T') \;=\; \min\{1, C_p(T')\}$$

where

$$C_p(T') \;:=\; \frac{p^{n-2}}{p^{n-2}-1}\frac{\beta_p^{\mathrm{Good}\cup Bad}(p^2T')}{\beta_p(T')}.$$

This yields

$$\begin{aligned} C_2(T') &= \frac{4}{3}\left(\frac{\beta_2^{\mathrm{Good}\cup Bad}(4T')}{\beta_2(T')}\right) \\ C_7(T') &= \frac{49}{48}\left(\frac{\beta_7^{\mathrm{Good}\cup Bad}(49T')}{\beta_7(T')}\right) \end{aligned}$$

Thus

Table 3.3: Local Information at 2

T'	$\beta_2(T')$	$\beta_2^{\text{Good}\cup Bad}(4T')$	$C_2(T')$
$1, 5$	$3/2$	$3/4$	$2/3$
$-1, -5$	$1/2$	$3/4$	2
$2, 10$	$5/4$	$3/4$	$4/5$
$-2, -10$	$3/4$	$3/4$	$4/3$

and

Table 3.4: Local Information at 7

T'	$\beta_7(T')$	$\beta_7^{G\cup B}(49T')$	$C_7(T')$
1	$6/7$	$400/7^3$	$25/21$
3	$8/7$	$2400/2401$	$25/28$
7	$342/7^3$	$2400/2401$	$175/171$
21	$2392/7^4$	$2400/2401$	$1225/1196$

Therefore, $\min_{T'}\left\{\prod_{p|N} C'_p(T')\right\} = \frac{2}{3}\cdot\frac{25}{28} = \frac{25}{42}$ and $Q = \langle 1, 1, 1, 7\rangle$ represents any m satisfying

$$\sqrt{(m)}/\tau(m)\prod_{p\nmid N, p|m, \chi(p)=-1}\frac{p-1}{p+1} > \frac{8\sqrt{14}}{7}\cdot\frac{42}{25} \approx 7.18398218260597.$$

The proof that Q is universal has thus been reduced to checking $r_Q(m) > 0$ for finitely-many m (i.e., those m which do not satisfy the above inequality).

Let $B(m) := \frac{\sqrt{m}}{\tau(m)}\prod_{p|m, \chi(p)=-1}\frac{p-1}{p+1}$. We say that a prime p is **eligible** if

$$B(p) < \frac{48\sqrt{14}}{25}\cdot\frac{1}{B(2)B(3)B(5)B(7)}.$$

Below is a table of eligible primes, listed by increasing $B(p)$:

Table 3.5: $B(p)$ values

p	$B(p)$	p	$B(p)$	p	$B(p)$	p	$B(p)$
2	.70711	103	5.07445	241	7.69794	401	10.01249
5	.74536	107	5.07626	251	7.92149	409	10.06255
3	.86603	109	5.22015	257	7.95347	419	10.23474
7	1.32288	113	5.31507	263	8.04721	421	10.25914
11	1.38193	127	5.54667	269	8.13986	431	10.33221
13	1.54524	131	5.72276	271	8.23104	433	10.35638
17	1.83249	137	5.85235	277	8.32166	439	10.47616
19	2.17945	139	5.89491	281	8.38153	443	10.47638
23	2.19809	151	6.06326	283	8.41130	449	10.59481
29	2.69258	149	6.10328	293	8.50040	457	10.68878
31	2.78388	157	6.18568	307	8.76071	461	10.68898
37	3.04138	163	6.30572	313	8.78956	463	10.71234
41	3.04911	167	6.46142	311	8.81760	467	10.80509
43	3.12969	173	6.50088	317	8.90225	479	10.94303
47	3.42783	179	6.61522	331	9.04190	487	10.98882
53	3.64004	181	6.65289	337	9.17878	491	11.03422
61	3.77915	191	6.83816	347	9.26044	499	11.12448
59	3.84057	193	6.94622	349	9.28740	503	11.21383
67	3.97230	197	7.01783	353	9.34107	509	11.23628
71	4.09605	199	7.05337	359	9.42102	521	11.36899
73	4.15654	211	7.19440	367	9.57862	523	11.43460
79	4.33299	223	7.46659	373	9.65660	541	11.62970
83	4.55522	229	7.50058	379	9.68273	547	11.65134
89	4.61217	227	7.53326	383	9.78519	557	11.80042
97	4.82393	233	7.63217	389	9.86154	563	11.86381
101	4.92641	239	7.66540	397	9.91237		

We say a positive (squarefree) integer m is **eligible** if $B(m) < \frac{48\sqrt{14}}{25}$. Note that it is not necessarily true that an eligible prime is an eligible number; however, all eligible numbers are divisible only by eligible primes. We now use Sage ([40]) to compile the list of all square-free eligible numbers.

Let $S(N)$ denote the number of eligible numbers which are the product of N distinct primes. Then

Table 3.7: $S(N)$ values

N	1	2	3	4	5	6	7	8	≥ 9
$S(N)$	46	294	819	1293	1221	600	134	5	0

The set of all eligible numbers therefore has cardinality 4469, and it is these numbers which we must check are represented by Q to confirm universality. As the largest eligible number is 15825810, we use the following low and cunning trick.

Consider the subform $R := \langle 1, 1, 1\rangle$ of Q. It is a classical result of Gauss that m is not represented by R if and only if $m = 4^k(8\ell + 7)$ for $k, \ell \in \mathbb{N} \cup \{0\}$. Given an eligible number m, then, we need only check $m - 7n^2$ (for some $n \in \mathbb{N} \cup \{0\}$) is represented by $\langle 1, 1, 1\rangle$.

Table 3.9: Exceptions to $\langle 1, 1, 1 \rangle$

n	# of eligible m with $m - 7n^2$ not represented by R	largest such m
0	349	1130415
1	29	220935
2	0	0

This then proves the positive universality of $Q = \langle 1, 1, 1, 7 \rangle$.

Chapter 4

Quadratic Forms over $\mathbb{Z}\left[(1+\sqrt{5})/2\right]$

4.1 Summary

This chapter concerns the values represented by two forms over $\mathbb{Q}(\sqrt{5})$: $\langle 1,1,1,1\rangle$ and $\langle 1,1,3,3(3+\sqrt{5})\rangle$. The sum of four squares is very well studied, and its universality is a consequence of [7] which shows the sum of three squares is universal over $\mathbb{Q}(\sqrt{5})$. Here, we derive an explicit formula for $r_{\langle 1,1,1,1\rangle}(m)$ written in such a way to highlight the parallels with the same formula over $\mathbb{Q}$. The original formula is due to Götzky in 1928 [19]. Predating Siegel, this paper does not derive the result via local densities. The form $\langle 1,1,3,3(3+\sqrt{5})\rangle$ has not been studied previously. We begin by showing that while all $m \in \mathcal{O}^+_{\mathbb{Q}(\sqrt{5})}$ are locally represented, globally the form is not universal. We then provide an explicit formula for the Eisenstein component of the theta series; necessary for this was computing a special value of an L-function over $\mathbb{Q}(\sqrt{5})$ with nontrivial character.

4.2 $\langle 1,1,1,1\rangle$

Theorem 4.2.1. *(Götzky, 1928, [19, Satz 4]) The form $Q(\vec{x}) = x_1^2+x_2^2+x_3^2+x_4^2$ is positive universal over $K=\mathbb{Q}(\sqrt{5})$. That is, for all $m \in \mathcal{O}_K^+$, $Q(\vec{x})=m$ has a solution in $\mathcal{O}_K^4$. Most specifically,*

$$r_Q(m) = 8\sum_{0\neq(d)|m} N(d) - 4\sum_{2|(d)|m} N(d) + 8\sum_{4|(d)|m} N(d).$$

Introductory Remarks

This form has discriminant $D_Q = D = 1$ and level $N_Q = N = 4$. The character is trivial as it is given by

$$\chi_Q(\mathfrak{p}) = \left(\frac{D}{\mathfrak{p}}\right) = \left(\frac{1}{\mathfrak{p}}\right) = 1.$$

We first check locally that quotients of $m \in \mathcal{O}_K^+$ by a square factor are represented mod $(2)^6$ and mod $\mathfrak{p}$ for all odd primes $\mathfrak{p}$. That all m are locally represented mod $\mathfrak{p}$ for all odd primes $\mathfrak{p}$ comes from the fact that over a finite field of odd characteristic the sum of two squares is universal. Verifying that all quotients of m by a square factor are represented mod $(2)^6$ is a relatively quick Sage computation.

The Eisenstein Coefficients

Theorem 4.2.2.

$$a_E(m) = \begin{cases} 8 \cdot \left(\sum\limits_{(0)\neq(d)|(m)} N(d) \right), & (2) \nmid (m) \\ 8 \cdot 3 \cdot \left(1 + 10 \sum\limits_{i=0}^{N-1} 4^{2(N-i+1)+1} \right) \cdot \left(\sum\limits_{(0)\neq(d)|(m),(2)\nmid(d)} N(d) \right), & \mathrm{ord}_{(2)}(m) = 2N+1, N \geq 0 \\ 8 \cdot 3 \cdot \left(9 + 10 \sum\limits_{i=0}^{N-2} 4^{2(N-i+1)} \right) \cdot \left(\sum\limits_{(0)\neq(d)|(m),(2)\nmid(d)} N(d) \right), & \mathrm{ord}_{(2)}(m) = 2N, N \geq 1. \end{cases}$$

Proof. We have by [38, pg. 285]

$$\begin{aligned} a_E(m) &= \prod_v \beta_v(m) \\ &= \left(\prod_{v|\infty} \beta_v(m) \right) \left(\beta_{(2)}(m) \right) \left(\prod_{(2)\neq\mathfrak{p}|(m)} \beta_{\mathfrak{p}}(m) \right) \left(\prod_{\mathfrak{p}\nmid(2m)} \beta_{\mathfrak{p}}(m) \right). \end{aligned}$$

In order to complete this proof we use a sequence of lemmas which we use to compute separately:

- $\prod\limits_{v|\infty} \beta_v(m)$

- $\prod_{\mathfrak{p}\nmid(2m)} \beta_{\mathfrak{p}}(m)$ (i.e., the local densities at finite primes dividing neither the level nor (m))
- $\beta_{(2)}(m)$ (the local density at all primes dividing the level)
- $\prod_{(2)\neq\mathfrak{p}\mid(m)} \beta_{\mathfrak{p}}(m)$ (the local density at all remaining primes).

Lemma 4.2.1. *For $Q = \langle 1,1,1,1\rangle$ and $m \in \mathcal{O}_K^+$,*

$$\prod_{v\mid\infty} \beta_v(m) = \frac{\pi^4 N(m)}{5^{3/2}}.$$

Proof. This is a direct application of [38, Hilfssatz 72]. □

Lemma 4.2.2. *For an odd prime $\mathfrak{p} \nmid (2m)$,*

$$\beta_{\mathfrak{p}}(m) = 1 - \frac{1}{N(\mathfrak{p})^2}.$$

Proof. For such a prime, the solutions are all of Good type and we have

$$\begin{aligned}
\beta_{\mathfrak{p}}(m) &= \lim_{v\to\infty} \frac{r_{\mathfrak{p}^v}^{\mathrm{Good}}(m)}{N(\mathfrak{p})^{3v}} \\
&= \lim_{v\to\infty} \frac{N(\mathfrak{p})^{3(v-1)} r_{\mathfrak{p}}^{\mathrm{Good}}(m)}{N(\mathfrak{p})^{3v}} \\
&= \frac{N(\mathfrak{p})^3 - N(\mathfrak{p})}{N(\mathfrak{p})^3} \\
&= 1 - \frac{1}{N(\mathfrak{p})^2}.
\end{aligned}$$

□

Thus,

$$\begin{aligned}
a_E(m) &= \left(\prod_{\nu|\infty}\beta_\nu(m)\right)\beta_{(2)}(m)\left(\prod_{(2)\neq\mathfrak{p}|m}\beta_\mathfrak{p}(m)\right)\left(\prod_{\mathfrak{q}\nmid 2m}\beta_\mathfrak{q}(m)\right)\\
&= \left(\frac{\pi^4 N(m)}{5^{3/2}}\right)\beta_{(2)}(m)\left(\prod_{(2)\neq\mathfrak{p}|m}\beta_\mathfrak{p}(m)\right)\frac{\prod_{\mathfrak{p}\nmid\infty}\left(1-\frac{1}{N(\mathfrak{p})^2}\right)}{\prod_{\mathfrak{p}|(2m)}\left(1-\frac{1}{N(\mathfrak{p})^2}\right)}\\
&= \left(\frac{\pi^4 N(m)}{5^{3/2}}\right)\beta_{(2)}(m)\left(\prod_{(2)\neq\mathfrak{p}|m}\beta_\mathfrak{p}(m)\right)\frac{1/\zeta_{\mathbb{Q}(\sqrt{5})}(2)}{\prod_{\mathfrak{p}|(2m)}\left(1-\frac{1}{N(\mathfrak{p})^2}\right)}\\
&= \frac{16\pi^4 N(m)}{15\cdot 5^{3/2}\zeta_{\mathbb{Q}(\sqrt{5})}(2)}\cdot\beta_{(2)}(m)\cdot\left(\prod_{(2)\neq\mathfrak{p}|m}\frac{N(\mathfrak{p})^2\beta_\mathfrak{p}(m)}{N(\mathfrak{p})^2-1}\right).
\end{aligned}$$

Lemma 4.2.3. $\zeta_{\mathbb{Q}(\sqrt{5})}(2) = \dfrac{2\pi^4}{75\sqrt{5}}$.

Proof.

$$\zeta_{\mathbb{Q}(\sqrt{5})}(2) = \zeta_\mathbb{Q}(2)L_\mathbb{Q}(\chi_5,2),$$

where χ_5 is the even, non-trivial (quadratic) character of conductor 5. As we have seen earlier, $\zeta_\mathbb{Q}(2) = \pi^2/6$. To compute $L_\mathbb{Q}(\chi_5,2)$, we have (using the notation of [23, pg. 104]):

$$L_\mathbb{Q}(\chi_5,2) = -\frac{2\pi^2}{25}\tau(\chi_5)L_\mathbb{Q}(1-2,\overline{\chi_5}),$$

where

$$\begin{aligned}
\tau(\chi_5) &= \sum_{a=1}^{5}\chi_5(a)e^{2\pi ia/5}\\
&= 2\left(\cos(2\pi/5)+\cos(\pi/5)\right)\\
&= \sqrt{5}
\end{aligned}$$

and

$$\begin{aligned} L_{\mathbb{Q}}(1-2,\overline{\chi_5}) &= L_{\mathbb{Q}}(1-2,\chi_5) \\ &= -\frac{1}{2}B_{2,\chi_5} \\ &= -\frac{1}{2}\left(\frac{1}{5}\sum_{a=1}^{5}(5)^2\chi_5(a)\left(\left(\frac{a-5}{5}\right)^2+\left(\frac{a-5}{5}\right)+\frac{1}{6}\right)\right) \\ &= -\frac{5}{2}\cdot\frac{4}{25} = -\frac{2}{5}. \end{aligned}$$

Hence

$$\begin{aligned} L_{\mathbb{Q}}(\chi_5,2) &= -\frac{2\pi^2}{25}\cdot\sqrt{5}\cdot\left(-\frac{2}{5}\right) \\ &= \frac{4\sqrt{5}\pi^2}{125} \end{aligned}$$

and

$$\zeta_{\mathbb{Q}(\sqrt{5})}(2) = \frac{\pi^2}{6}\cdot\frac{4\sqrt{5}\pi^2}{125} = \frac{2\pi^4}{75\sqrt{5}}$$

as claimed. □

Hence

$$\begin{aligned} a_E(m) &= \frac{16\pi^4 N(m)}{15\cdot 5^{3/2}\zeta_{\mathbb{Q}(\sqrt{5})}(2)}\cdot\beta_{(2)}(m)\cdot\left(\prod_{(2)\neq\mathfrak{p}|m}\frac{N(\mathfrak{p})^2\beta_{\mathfrak{p}}(m)}{N(\mathfrak{p})^2-1}\right) \\ &= 8N(m)\beta_{(2)}(m)\left(\prod_{\mathfrak{p}|m,\mathfrak{p}\nmid 2}\frac{N(\mathfrak{p})^2\beta_{\mathfrak{p}}(m)}{N(\mathfrak{p})^2-1}\right). \end{aligned}$$

Lemma 4.2.4.

$$\beta_{(2)}(m) = \begin{cases} 1, & (2) \nmid (m) \\ \frac{15}{8}\left(\sum_{i=0}^{N-1} \frac{1}{4^{2i}}\right) + \frac{3}{4^{2N+1}}, & \mathrm{ord}_{(2)}(m) = 2N+1, N \geq 0 \\ \frac{15}{8}\left(\sum_{i=0}^{N-2} \frac{1}{4^{2i}}\right) + \frac{27}{4^{2N}}, & \mathrm{ord}_{(2)}(m) = 2N, N > 0. \end{cases}$$

Proof. When $(2) \nmid (m)$, all solutions are of Good type and we have:

$$\begin{aligned} \beta_{(2)}(m) &= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{(2)^v}(m)}{4^{3v}} \\ &= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{(2)^{3+(v-3)}}(m)}{4^{3v}} \\ &= \lim_{v\to\infty} \frac{4^{3(v-3)} r^{\mathrm{Good}}_{(8)}(m)}{4^{3v}} \\ &= \frac{262144}{2^{18}} = 1. \end{aligned}$$

For $\mathrm{ord}_2(m) = 2N+1$, $N \geq 0$, we have Good and Zero type solutions with

$$\begin{aligned} \beta_{(2)}(m) &= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{(2)^v}(m)}{4^{3v}} + \lim_{v\to\infty} \frac{r^{\mathrm{Zero}}_{(2)^v}(m)}{4^{3v}} \\ &= \lim_{v\to\infty} \frac{4^{3(v-3)} r^{\mathrm{Good}}_{(8)}(m)}{4^{3v}} + \lim_{v\to\infty} \frac{1}{4^{3v}} \left(\sum_{i=1}^{N} 4^{4i} r^{\mathrm{Good}}_{(2)^{v-2i}}\left(\frac{m}{2^{2i}}\right)\right) \\ &= \frac{2^{15}\cdot 3\cdot 5}{2^{18}} + \frac{1}{4^9}\left(\sum_{i=1}^{N} 4^{-2i} r^{\mathrm{Good}}_{(8)}\left(\frac{m}{2^{2i}}\right)\right) \\ &= \frac{15}{8} + \frac{\sum_{i=1}^{N-1} 2^{15}\cdot 3\cdot 5}{2^{18}\cdot 4^{2i}} + \frac{2^{16}\cdot 3}{4^9\cdot 4^{2N}} \\ &= \frac{15}{8}\left(\sum_{i=0}^{N-1} \frac{1}{4^{2i}}\right) + \frac{3}{4^{2N+1}}. \end{aligned}$$

Last, when $\mathrm{ord}_2(m) = 2N$, $N \geq 1$, we have Good and Zero type solutions, and

$$\begin{aligned}
\beta_{(2)}(m) &= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{(2)^v}(m)}{4^{3v}} + \lim_{v\to\infty} \frac{r^{\mathrm{Zero}}_{(2)^v}(m)}{4^{3v}} \\
&= \frac{15}{8} + \lim_{v\to\infty} \frac{1}{4^{3v}} \left(\sum_{i=1}^{N} 4^{4i} r^{\mathrm{Good}}_{(2)^{v-2i}} \left(\frac{m}{2^{2i}} \right) \right) \\
&= \frac{15}{8} + \frac{1}{4^9} \left(\sum_{i=1}^{N} 4^{-2i} r^{\mathrm{Good}}_{(8)} \left(\frac{m}{2^{2i}} \right) \right) \\
&= \frac{15}{8} + \sum_{i=1}^{N-2} \frac{r^{\mathrm{Good}}_{(8)} \left(\frac{m}{2^{2i}} \right)}{4^{2i+9}} + \frac{r^{\mathrm{Good}}_{(8)} \left(\frac{m}{2^{2(N-1)}} \right)}{4^{9+2(N-1)}} + \frac{r^{\mathrm{Good}}_{(8)} \left(\frac{m}{2^{2N}} \right)}{4^{2N+9}} \\
&= \frac{15}{8} + \frac{15}{8} \left(\sum_{i=1}^{N-2} 4^{-2i} \right) + \frac{2^{15} \cdot 13}{2^{18} \cdot 4^{2(N-1)}} + \frac{2^{18}}{2^{18} \cdot 4^{2N}} \\
&= \frac{15}{8} \left(\sum_{i=0}^{N-2} \frac{1}{4^{2i}} \right) + \frac{13}{8} \frac{1}{4^{2(N-1)}} + \frac{1}{4^{2N}} \\
&= \frac{15}{8} \left(\sum_{i=0}^{N-2} \frac{1}{4^{2i}} \right) + \frac{27}{4^{2N}}.
\end{aligned}$$

□

Lemma 4.2.5. *For* $\mathfrak{p} \neq (2)$, $\mathfrak{p}|(m)$

$$\frac{\beta_{\mathfrak{p}}(m) N(\mathfrak{p})^2}{N(\mathfrak{p})^2 - 1} = \sum_{i=0}^{\mathrm{ord}_{\mathfrak{p}}(m)} N(\mathfrak{p})^{-i}.$$

Proof. Suppose $\mathrm{ord}_{\mathfrak{p}}(m) = 2N$, $N \in \mathbb{N}$. Here both Good and Zero type solutions exist and

$$\begin{aligned}
\beta_{\mathfrak{p}}(m) &= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}} + \lim_{v\to\infty} \frac{r^{\mathrm{Zero}}_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}} \\
&= \frac{N(\mathfrak{p})^3 + N(\mathfrak{p})(N(\mathfrak{p}) - 1) - 1}{N(\mathfrak{p})^3} \\
&\quad + \lim_{v\to\infty} \frac{1}{N(\mathfrak{p})^{3v}} \left(\left(\sum_{i=1}^{N-1} N(\mathfrak{p})^{4i} r^{\mathrm{Good}}_{\mathfrak{p}^{v-2i}}(m/\mathfrak{p}^{2i}) \right) + N(\mathfrak{p})^{4N} r^{\mathrm{Good}}_{\mathfrak{p}^{v-2N}}(m/\mathfrak{p}^{2N}) \right) \\
&= \left(\sum_{i=0}^{N-1} N(\mathfrak{p})^{-2i} \right) \left(\frac{N(\mathfrak{p})^3 + N(\mathfrak{p})(N(\mathfrak{p}) - 1) - 1}{N(\mathfrak{p})^3} \right) + N(\mathfrak{p})^{-2N} \left(1 - 1/N(\mathfrak{p})^2 \right).
\end{aligned}$$

So for such primes

$$\begin{aligned}\frac{\beta_{\mathfrak{p}}(m)N(\mathfrak{p})^2}{N(\mathfrak{p})^2-1} &= \left(\sum_{i=0}^{N-1} N(\mathfrak{p})^{-2i}\right)\left(1+\frac{1}{N(\mathfrak{p})}\right)+N(\mathfrak{p})^{-2N} \\ &= \sum_{i=0}^{2N} N(\mathfrak{p})^{-i}.\end{aligned}$$

Last, suppose $\mathrm{ord}_{\mathfrak{p}}(m) = 2N+1$, $N \in \mathbb{N}\cup\{0\}$. Again only Zero and Good type solutions exist with

$$\begin{aligned}\beta_{\mathfrak{p}}(m) &= \lim_{v\to\infty}\frac{r_{\mathfrak{p}^v}^{\mathrm{Good}}(m)}{N(\mathfrak{p})^{3v}}+\lim_{v\to\infty}\frac{r_{\mathfrak{p}^v}^{\mathrm{Zero}}(m)}{N(\mathfrak{p})^{3v}} \\ &= \frac{N(\mathfrak{p})^3+N(\mathfrak{p})(N(\mathfrak{p})-1)-1}{N(\mathfrak{p})^3}+\lim_{v\to\infty}\frac{1}{N(\mathfrak{p})^{3v}}\left(\sum_{i=1}^{N} N(\mathfrak{p})^{4i} r_{\mathfrak{p}^{v-2i}}^{\mathrm{Good}}(m/\mathfrak{p}^{2i})\right) \\ &= \left(\sum_{i=0}^{N} N(\mathfrak{p})^{-2i}\right)\left(\frac{N(\mathfrak{p})^3+N(\mathfrak{p})(N(\mathfrak{p})-1)-1}{N(\mathfrak{p})^3}\right).\end{aligned}$$

Hence for such primes

$$\begin{aligned}\frac{\beta_{\mathfrak{p}}(m)N(\mathfrak{p})^2}{N(\mathfrak{p})^2-1} &= \left(\sum_{i=0}^{N} N(\mathfrak{p})^{-2i}\right)\left(1+\frac{1}{N(\mathfrak{p})}\right) \\ &= \sum_{i=0}^{2N+1} N(\mathfrak{p})^{-i}.\end{aligned}$$

□

In conclusion, for m odd, we see

$$\begin{aligned} a_E(m) &= 8N(m)\beta_{(2)}(m)\left(\prod_{(2)\neq\mathfrak{p}|(m)}\frac{N(\mathfrak{p})^2\beta_{\mathfrak{p}}(m)}{N(\mathfrak{p})^2-1}\right) \\ &= 8N(m)\cdot 1\cdot\left(\prod_{\mathfrak{p}|(m)}\left(\sum_{i=0}^{\mathrm{ord}_{\mathfrak{p}}(m)}N(\mathfrak{p})^{-i}\right)\right) \\ &= 8\prod_{\mathfrak{p}|(m)}\left(\sum_{i=0}^{\mathrm{ord}_{\mathfrak{p}}(m)}N(\mathfrak{p})^{i}\right) \\ &= 8\cdot\left(\sum_{(0)\neq(d)|(m)}N(d)\right). \end{aligned}$$

Similarly when $\mathrm{ord}_{(2)}(m) = 2N+1$, $N \geq 0$

$$\begin{aligned} a_E(m) &= 8N(m)\beta_{(2)}(m)\left(\prod_{(2)\neq\mathfrak{p}|(m)}\frac{N(\mathfrak{p})^2\beta_{\mathfrak{p}}(m)}{N(\mathfrak{p})^2-1}\right) \\ &= 8N(m)\left(\frac{15}{8}\left(\sum_{i=0}^{N-1}\frac{1}{4^{2i}}\right)+\frac{3}{4^{2N+1}}\right)\left(\prod_{(2)\neq\mathfrak{p}|(m)}\left(\sum_{i=0}^{\mathrm{ord}_{\mathfrak{p}}(m)}N(\mathfrak{p})^{-i}\right)\right) \\ &= 8\left(3+\frac{15}{8}\sum_{i=0}^{N-1}4^{2(N-i)+1}\right)\cdot\left(\prod_{(2)\neq\mathfrak{p}|(m)}\sum_{i=0}^{\mathrm{ord}_{\mathfrak{p}}(m)}N(\mathfrak{p})^{i}\right) \\ &= 8\cdot 3\cdot\left(1+10\sum_{i=0}^{N-1}4^{2(N-i+1)+1}\right)\cdot\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)}N(d)\right) \end{aligned}$$

and when $ord_{(2)}(m) = 2N$, $N \geq 1$

$$\begin{aligned}
a_E(m) &= 8N(m)\beta_{(2)}(m)\left(\prod_{(2)\neq\mathfrak{p}|(m)} \frac{N(\mathfrak{p})^2\beta_{\mathfrak{p}}(m)}{N(\mathfrak{p})^2-1}\right) \\
&= 8N(m)\left(\frac{15}{8}\left(\sum_{i=0}^{N-2}\frac{1}{4^{2i}}\right)+\frac{27}{4^{2N}}\right)\left(\prod_{(2)\neq\mathfrak{p}|(m)}\left(\sum_{i=0}^{\mathrm{ord}_{\mathfrak{p}}(m)} N(\mathfrak{p})^{-i}\right)\right) \\
&= 8\left(27+\frac{15}{8}\sum_{i=0}^{N-2}4^{2(N-i)}\right)\cdot\left(\prod_{(2)\neq\mathfrak{p}|(m)}\sum_{i=0}^{\mathrm{ord}_{\mathfrak{p}}(m)} N(\mathfrak{p})^{i}\right) \\
&= 8\cdot 3\cdot\left(9+10\sum_{i=0}^{N-2}4^{2(N-i+1)}\right)\cdot\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right).
\end{aligned}$$

This completes the proof of the theorem. □

Explicit Representation Results

Lemma 4.2.6. *For $Q = \langle 1,1,1,1\rangle$ over $\mathbb{Q}(\sqrt{5})$, $r_Q(m) \equiv a_E(m)$ for all $m \in \mathcal{O}_K^+$.*

Proof. It is known that the space of Hilbert modular forms of parallel weight two and level 4 has a trivial cuspidal space. Thus for all m, $a_C(m) = 0$ and $r_Q(m) = a_E(m)$ [10, Example 1]. One can also verify this using the code included in Appendix A.2 which implements Dembélé's algorithm over $\mathbb{Q}(\sqrt{5})$ and trivial character. That the dimension of the cusp space is 0, however, completes the proof of the lemma. □

The lemma does not just imply the universality of $Q = \langle 1,1,1,1\rangle$ over $\mathbb{Q}(\sqrt{5})$. It also proves

$$\begin{aligned}
r_Q(m) &= a_E(m) \\
&= \begin{cases}
8\cdot\left(\sum_{(0)\neq(d)|(m)} N(d)\right), & (2)\nmid(m) \\
8\cdot 3\cdot\left(1+10\sum_{i=0}^{N-1}4^{2(N-i+1)+1}\right)\cdot\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right), & \mathrm{ord}_{(2)}(m) = 2N+1, N\geq 0 \\
8\cdot 3\cdot\left(9+10\sum_{i=0}^{N-2}4^{2(N-i+1)}\right)\cdot\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right), & \mathrm{ord}_{(2)}(m) = 2N, N\geq 1.
\end{cases}
\end{aligned}$$

To highlight these representation results, we return to an example from a previous chapter of code output:

```
sage: x = var('x')
sage: K.<a>=NumberField(x^2-5,embedding=1)
sage: S = Matrix(K,4,[1,0,0,0,  0,1,0,0,  0,0,1,0,  0,0,0,1])
sage: List_for_all_Theta(5,S,5)
[(1, 0), (1, 1), (2, -1), (2, 0), (2, 1), (2, 2), (2, 3), (3, -1), (3, 0),
(3, 1), (3, 2), (3, 3), (3, 4), (4, -2), (4, -1), (4, 0),
(4, 1), (4, 2), (5, -3), (5, -2), (5, -1), (5, 0), (6, -3), (6, -2), (7, -4)]
sage: Theta_Computation(5,S,5)
[8, 8, 8, 24, 48, 24, 8, 48, 80,
96, 96, 80, 48, 24, 96, 216,
160, 144, 8, 96, 160, 248, 80, 144, 48]
```

This code concerns representation by the sum of four squares over $\mathbb{Q}(\sqrt{5})$, returning a list of $r_Q(\ell)$ for ℓ depending upon $m = 5$. We compare the code outputs to the outputs of the explicit local density formula for select elements:

Table 4.1: Comparison of Representation Numbers

$\ell = (a, b) \leftrightarrow a + b\theta$	$N(\ell)$	$ord_{(2)}(m)$	Code Output	Representation Theorem
(1,0)	1	0	8	8
(1,1)	1	0	8	8
(2,0)	4	1	24	$8 \cdot 3 \cdot 1$
(2,1)	5	0	48	$8 \cdot (1 + 5)$
(4,0)	16	2	216	$8 \cdot 3 \cdot 9$
(6,-2)	20	1	144	$8 \cdot 3 \cdot 1 \cdot (5 + 1)$

Recovering Götzky's Result

What remains is to derive Götzky's result. That is, knowing now that

$$\begin{aligned} r_Q(m) &= a_E(m) \\ &= \begin{cases} 8 \cdot \left(\sum_{(0)\neq(d)|(m)} N(d) \right), & (2) \nmid (m) \\ 8 \cdot 3 \cdot \left(1 + 10 \sum_{i=0}^{N-1} 4^{2(N-i+1)+1} \right) \cdot \left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d) \right), & ord_{(2)}(m) = 2N+1, N \geq 0 \\ 8 \cdot 3 \cdot \left(9 + 10 \sum_{i=0}^{N-2} 4^{2(N-i+1)} \right) \cdot \left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d) \right), & ord_{(2)}(m) = 2N, N \geq 1. \end{cases} \end{aligned}$$

we want to show that that simplifies to

$$r_Q(m) = 8 \sum_{0\neq(d)|m} N(d) - 4 \sum_{2|(d)|m} N(d) + 8 \sum_{4|(d)|m} N(d).$$

We proceed by cases. When $\mathrm{ord}_{(2)}(m) = 0$, then certainly

$$8 \cdot \left(\sum_{(0)\neq(d)|(m)} N(d) \right) = 8 \sum_{0\neq(d)|m} N(d) - 4 \sum_{2|(d)|m} N(d) + 8 \sum_{4|(d)|m} N(d).$$

For the remaining two cases, we write Götzky's result in the following way:

$$\begin{aligned} 8 \sum_{0\neq(d)|m} N(d) - 4 \sum_{2|(d)|m} N(d) + 8 \sum_{4|(d)|m} N(d) &= 8 \sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)=0} N(d) + 4 \sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d) \\ &\quad + 12 \sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)\geq 2} N(d). \end{aligned}$$

We now note

$$\begin{aligned} 8 \cdot 3 \cdot \left(1 + 10 \sum_{i=0}^{N-1} 4^{2(N-i+1)+1} \right) \cdot \left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d) \right) &= 8 \left(1 + 2 + \frac{15}{8} \sum_{i=0}^{N-1} 4^{2(N-i)+1} \right) \\ &\quad \cdot \left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d) \right). \end{aligned}$$

Allowing $\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right) = S$, this gives:

$$\begin{aligned}
8\left(1+2+\frac{15}{8}\sum_{i=0}^{N-1}4^{2(N-i)+1}\right)S &= 8S+16S+15S\sum_{i=0}^{N-1}4^{2(N-i)+1} \\
&= 8S+4\sum_{(0)(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d)+3(4+1)S\sum_{i=0}^{N-1}4^{2(N-i)+1} \\
&= 8S+4\sum_{(0)(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d)+3S\cdot 4\sum_{i=2}^{2N+1}4^{i} \\
&= 8S+4\sum_{(0)(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d)+12\sum_{(0)\neq(d)|m,\mathrm{ord}_2(d)\geq 2} N(d).
\end{aligned}$$

Götzky's result then holds.

Finally, we suppose $\mathrm{ord}_{(2)}(m) = 2N$, $N \in \mathbb{N}$. Then we want to show again

$$\begin{aligned}
8\cdot 3\cdot\left(9+10\sum_{i=0}^{N-2}4^{2(N-i+1)}\right)\cdot\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right) &= 8\sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)=0} N(d) \\
&\quad +4\sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d) \\
&\quad +12\sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)\geq 2} N(d).
\end{aligned}$$

Again, we begin by seeing

$$8\cdot 3\cdot\left(9+10\sum_{i=0}^{N-2}4^{2(N-i+1)}\right)\cdot\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right) = 8\left(27+\frac{15}{8}\sum_{i=0}^{N-2}4^{2(N-i)}\right)\cdot\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right)$$

Again, allowing $\left(\sum_{(0)\neq(d)|(m),(2)\nmid(d)} N(d)\right) = S$ we get:

$$\begin{aligned}
8\left(27+\frac{15}{8}\sum_{i=0}^{N-2}4^{2(N-i)}\right)S &= 8S+8\left(26+\frac{15}{8}\sum_{i=0}^{N-2}4^{2(N-i)}\right)S \\
&= 8S+\left(16\cdot 13+15\sum_{i=0}^{N-2}4^{2(N-i)}\right)S \\
&= 8S+\left(16(1+12)+3(4+1)\sum_{i=1}^{N}4^{2i}\right)S \\
&= 8S+4\sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d)+\left(16\cdot 12+3(4+1)\sum_{i=1}^{N}4^{2i}\right)S \\
&= 8S+4\sum_{(0)\neq(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d)+12\left(\sum_{i=2}^{2N}4^{i}\right)S \\
&= 8S+4\sum_{(0)(d)|m,\mathrm{ord}_{(2)}(d)=1} N(d)+12\sum_{(0)\neq(d)|m,\mathrm{ord}_2(d)\geq 2} N(d).
\end{aligned}$$

This completes the analysis of the sum of four squares over $\mathbb{Z}[(1+\sqrt{5})/2]$.

4.3 $\langle 1,1,3,3(3+\sqrt{5})\rangle$

Theorem 4.3.1. *$Q=\langle 1,1,3,3(3+\sqrt{5})\rangle$ is not positive universal over $\mathcal{O}_K$ where $K=\mathbb{Q}(\sqrt{5})$.*

Proof. We will show for $m=3+2\left(\frac{1+\sqrt{5}}{2}\right)$, $Q(\vec{x})=m$ has no solution in $\mathcal{O}_K$. Suppose Q represents $3+2\left(\frac{1+\sqrt{5}}{2}\right)$. For $i=1,...,4$, set $x_i:=a_i+b_i\left(\frac{1+\sqrt{5}}{2}\right)$ for $a_i,b_i\in\mathbb{Z}$. Then

$$\begin{aligned}
Q(\vec{x}) &= x_1^2+x_2^2+3x_3^2+3(3+\sqrt{5})x_4^2 \\
&= \left(a_1^2+b_1^2+a_2^2+b_2^2+3(a_3^2+b_3^2)+3(3+\sqrt{5})(a_4^2+b_4^2)\right) \\
&\quad +\left(b_1^2+2b_1a_1+b_2^2+2b_2a_2+3(b_3^2+2b_3a_3)+3(3+\sqrt{5})(b_4^2+b_4a_4)\right)\left(\frac{1+\sqrt{5}}{2}\right).
\end{aligned}$$

In order to satisfy

$$3 = \left(a_1^2 + b_1^2 + a_2^2 + b_2^2 + 3(a_3^2 + b_3^2) + 3(3+\sqrt{5})(a_4^2 + b_4^2)\right)$$

we see immediately $a_4 = b_4 = 0$; moreover, for $i \neq 4$, $|a_i b_i| \leq 1$. If either $|a_3| = 1$ or $|b_3| = 1$, then we force $a_1, a_2, b_1, b_2 = 0$; however, then

$$2\left(\frac{1+\sqrt{5}}{2}\right) = 3(b_3^2 + 2b_3 a_3)\left(\frac{1+\sqrt{5}}{2}\right)$$

has no solution. Thus $a_3 = b_3 = 0$ and

$$3 + 2\left(\frac{1+\sqrt{5}}{2}\right) = \left(a_1^2 + b_1^2 + a_2^2 + b_2^2\right) + \left(b_1^2 + 2b_1 a_1 + b_2^2 + 2b_2 a_2\right)\left(\frac{1+\sqrt{5}}{2}\right).$$

In writing 3 as a sum of 4 integer squares, we see that exactly three of $a_1, b_1, a_2, b_2 = \pm 1$, and exactly one is 0. This forces $b_i^2 + 2b_i a_i \in \{-1, 0, 1, 3\}$. Without loss of generality, suppose $a_1 b_1 = 0$. This forces $b_1^2 + 2a_1 b_1 \in \{0, 1\}$, and $b_2^2 + 2a_2 b_2 \in \{-1, 3\}$. But then there is no solution to $b_1^2 + 2b_1 a_1 + b_2^2 + 2b_2 a_2 = 2$.

Thus there is no $\vec{x} \in \mathcal{O}_K^4$ with $Q(\vec{x}) = 3 + 2\left(\frac{1+\sqrt{5}}{2}\right)$. □

Introductory Remarks

The determinant is $\det(Q) = 9(3+\sqrt{5}) = 3^2 \cdot 2 \cdot ((1+\sqrt{5})/2)^2 = 18$. The level is $N_Q = (24) = (2)^3 \cdot (3)$. By definition,

$$\chi(\mathfrak{p}) = \left(\frac{D_Q}{\mathfrak{p}}\right) = \left(\frac{3^2 \cdot 2 \cdot ((1+\sqrt{5})/2)^2}{\mathfrak{p}}\right) = \left(\frac{2}{\mathfrak{p}}\right) = \begin{cases} 1, & N(\mathfrak{p}) \equiv \pm 1 \pmod 8 \\ -1, & N(\mathfrak{p}) \equiv \pm 3 \pmod 8. \end{cases}$$

The Eisenstein Coefficients

Siegel's Theorem on Local Densities, [38, pg. 285], says that

$$\begin{aligned} a_E(m) &= \prod_v \beta_v(m) \\ &= \left(\prod_{v|\infty} \beta_\infty(m)\right) \beta_{(2)}(m)\beta_{(3)}(m) \left(\prod_{(2),(3)\neq\mathfrak{p}|m} \beta_\mathfrak{p}(m)\right) \left(\prod_{\mathfrak{q}\nmid(6m)} \beta_\mathfrak{q}(m)\right). \end{aligned}$$

Again, we provide a sequence of lemmas which will completely determine $a_E(m)$. Namely, we consider separately:

- $\prod_{v|\infty} \beta_v(m)$
- $\prod_{\mathfrak{q}\nmid(6m)} \beta_\mathfrak{q}(m)$ (i.e., the local densities at all primes dividing neither the level nor (m))
- $\beta_{(2)}(m)\beta_{(3)}(m)$ (the local densities at the primes dividing the level)
- $\prod_{(2),(3)\neq\mathfrak{p}|m} \beta_\mathfrak{p}(m)$ (the local densities at all remaining primes).

Lemma 4.3.1. $\prod_{v|\infty} \beta_v(m) = \dfrac{\pi^4 N(m)}{18 \cdot 5^{3/2}}$.

Proof. This is a direct application of [38, Hilfssatz 72]. □

Lemma 4.3.2. *For all finite primes* $\mathfrak{q} \nmid (m)$ *and* $\mathfrak{q} \nmid N_Q$,

$$\beta_\mathfrak{q}(m) = 1 - \frac{\chi(\mathfrak{q})}{N(\mathfrak{q})^2}.$$

Proof. For these primes all solutions are Good with

$$\begin{aligned}
\beta_{\mathfrak{q}}(m) &= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{\mathfrak{q}^v}(m)}{N(\mathfrak{q})^{3v}} \\
&= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{\mathfrak{q}^{(v-1)+1}}(m)}{N(\mathfrak{q})^{3v}} \\
&= \lim_{v\to\infty} \frac{N(\mathfrak{q})^{3(v-1)} r^{\mathrm{Good}}_{\mathfrak{q}}(m)}{N(\mathfrak{q})^{3v}} \\
&= \frac{r_{\mathfrak{q}}(m)}{N(\mathfrak{q})^3} \\
&= \frac{N(\mathfrak{q})^3 - N(\mathfrak{q})\chi(\mathfrak{q})}{N(\mathfrak{q})^3} \\
&= 1 - \chi(\mathfrak{q})/N(\mathfrak{q})^2.
\end{aligned}$$

□

Thus

$$\begin{aligned}
\left(\prod_{2,3\neq\mathfrak{q}\nmid(m)} \beta_{\mathfrak{q}}(m)\right) &= \frac{\prod_{\mathfrak{p}\nmid\infty} 1-\chi(\mathfrak{p})/N(\mathfrak{p})^2}{\prod_{\mathfrak{p}|6m} 1-\chi(\mathfrak{p})/N(\mathfrak{p})^2} \\
&= \frac{1}{L_{\mathbb{Q}(\sqrt{5})}(2,\chi)} \left(\prod_{\mathfrak{p}|6m} \frac{N(\mathfrak{p})^2}{N(\mathfrak{p})^2-\chi(\mathfrak{p})}\right)
\end{aligned}$$

and

$$\begin{aligned}
a_E(m) &= \prod_v \beta_v(m) \\
&= \left(\prod_{v|\infty} \beta_\infty(m)\right) \beta_{(2)}(m)\beta_{(3)}(m) \left(\prod_{(2),(3)\neq\mathfrak{p}|m} \beta_{\mathfrak{p}}(m)\right) \left(\prod_{\mathfrak{q}\nmid(6m)} \beta_{\mathfrak{q}}(m)\right) \\
&= \frac{\pi^4 N(m)}{18\cdot 5^{3/2} L_{\mathbb{Q}(\sqrt{5})}(2,\chi)} \frac{\beta_{(2)}(m)N(2)^2}{N(2)^2-\chi(2)} \cdot \frac{\beta_{(3)}(m)N(3)^2}{N(3)^2-\chi(3)} \left(\prod_{(2),(3)\neq\mathfrak{p}|m} \frac{\beta_{\mathfrak{p}}(m)N(\mathfrak{p})^2}{N(\mathfrak{p})^2-1}\right).
\end{aligned}$$

Lemma 4.3.3.

$$\begin{aligned} L_{\mathbb{Q}(\sqrt{5})}(2,\chi) &= L_{\mathbb{Q}}(2,\psi)L_{\mathbb{Q}}(2,\psi\chi_5) \\ &= \frac{7\sqrt{5}\pi^4}{1600} \end{aligned}$$

where $\psi(p) = \left(\frac{2}{p}\right)$.

Proof. The first set of equalities is a special case of a more general statement given in the background chapter on L-functions.

To compute the value of the L-function, we now use the techniques outlined in [23, pg. 104]:

$$\begin{aligned} L_{\mathbb{Q}}(2,\psi) &= \frac{\tau(\psi)}{2}\left(\frac{2\pi}{8}\right)^2 \frac{L_{\mathbb{Q}}(1-2,\overline{\psi})}{\cos\pi} \\ &= -\frac{\pi^2}{32}\tau(\psi)L_{\mathbb{Q}}(1-2,\overline{\psi}) \end{aligned}$$

where

$$\begin{aligned} \tau(\psi) &:= \sum_{a=1}^{8}\psi(a)e^{2\pi ia/8} \\ &= 2\sqrt{2} \end{aligned}$$

and

$$\begin{aligned} L_{\mathbb{Q}}(1-2,\overline{\psi}) &= L_{\mathbb{Q}}(1-2,\psi) \\ &= -\frac{1}{2}\left(\frac{1}{8}\sum_{a=1}^{8}\psi(a)8^2B_2\left(\frac{a-8}{8}\right)\right) \\ &= -1. \end{aligned}$$

Thus

$$\begin{aligned} L_{\mathbb{Q}}(2,\psi) &= -\frac{\pi^2}{32}\tau(\psi)L_{\mathbb{Q}}(1-2,\psi) \\ &= -\frac{\pi^2}{32}\cdot 2\sqrt{2}\cdot -\frac{1}{2}\cdot 2 \\ &= \frac{\pi^2\sqrt{2}}{16}. \end{aligned}$$

Next, we consider $L_{\mathbb{Q}}(2,\psi\chi_5)$. Note that $\psi\chi_5$ has conductor 40 with

$$\psi\chi_5(p) = \begin{cases} 1, & p \equiv 1,3,9,13,27,31,37,39 \pmod{40} \\ -1, & p \equiv 7,11,17,19,21,23,29,33 \pmod{40} \end{cases}.$$

Again, by [23, pg. 104] we have:

$$\begin{aligned} L_{\mathbb{Q}}(2,\psi\chi_D) &= \frac{\tau(\psi\chi_D)}{2\cdot 1}\left(\frac{2\pi}{40}\right)^2 \cdot \frac{L_{\mathbb{Q}}(1-2,\overline{\psi\chi_D})}{\cos(\pi)} \\ &= -\frac{\tau(\psi\chi_D)}{2}\left(\frac{\pi^2}{400}\right) L_{\mathbb{Q}}(1-2,\overline{\psi\chi_D}). \end{aligned}$$

Again

$$\begin{aligned} L_{\mathbb{Q}}(1-2,\overline{\psi\chi_D}) &= L_{\mathbb{Q}}(1-2,\psi\chi_D) \\ &= -\frac{1}{2}B_{2,\psi\chi_D} \\ &= -20\sum_{a=1}^{40}\psi\chi_D(a)B_2\left(\frac{a-40}{40}\right) \\ &= -14. \end{aligned}$$

Additionally,

$$\begin{aligned} \tau(\psi\chi_D) &:= \sum_{a=1}^{40}\psi\chi_D(a)e^{2\pi ia/40} \\ &= 2\sqrt{10} \end{aligned}$$

and more importantly:

$$\begin{aligned} L_{\mathbb{Q}}(2,\psi\chi_D) &= \frac{-\tau(\psi\chi_D)}{2}\left(\frac{\pi^2}{400}\right)L_{\mathbb{Q}}(1-2,\overline{\psi\chi_5}) \\ &= -\sqrt{10}\left(\frac{\pi^2}{400}\right)\cdot(-14) \\ &= \frac{7\sqrt{10}\pi^2}{200}. \end{aligned}$$

Hence

$$\begin{aligned} L_{\mathbb{Q}(\sqrt{5})}(2,\chi) &= \frac{\sqrt{2}\pi^2}{16}\cdot\frac{7\sqrt{10}\pi^2}{200} \\ &= \frac{7\sqrt{5}\pi^4}{1600}. \end{aligned}$$

□

Our Eisenstein component is therefore

$$\begin{aligned} a_E(m) &= \prod_v \beta_v(m) \\ &= \beta_\infty(m)\beta_{(2)}(m)\beta_{(3)}(m)\left(\prod_{(2),(3)\neq\mathfrak{p}|m}\beta_{\mathfrak{p}}(m)\right)\left(\prod_{\mathfrak{q}\nmid 6m}\beta_{\mathfrak{q}}(m)\right) \\ &= \frac{\pi^4 N(m)}{18\cdot 5^{3/2}}\beta_{(2)}(m)\beta_{(3)}(m)\left(\prod_{(2),(3)\neq\mathfrak{p}|m}\beta_{\mathfrak{p}}(m)\right)\left(\frac{1}{L_{\mathbb{Q}(\sqrt{5})}(2,\chi)}\left(\prod_{\mathfrak{p}|6m}\frac{N(\mathfrak{p})^2}{N(\mathfrak{p})^2-\chi(\mathfrak{p})}\right)\right) \\ &= \frac{\pi^4 N(m)}{18\cdot 5^{3/2}}\cdot\frac{\beta_{(2)}(m)N(2)^2}{N(2)^2-\chi(2)}\cdot\frac{\beta_{(3)}(m)N(3)^2}{N(3)^2-\chi(3)}\cdot\frac{1}{L_{\mathbb{Q}(\sqrt{5})}(2,\chi)}\left(\prod_{(2),(3)\neq\mathfrak{p}|m}\frac{\beta_{\mathfrak{p}}(m)N(\mathfrak{p})^2}{N(\mathfrak{p})^2-\chi(\mathfrak{p})}\right) \\ &= \frac{32N(m)}{63}\cdot\frac{\beta_{(2)}(m)N(2)^2}{N(2)^2-\chi(2)}\cdot\frac{\beta_{(3)}(m)N(3)^2}{N(3)^2-\chi(3)}\left(\prod_{(2),(3)\neq\mathfrak{p}|m}\frac{\beta_{\mathfrak{p}}(m)N(\mathfrak{p})^2}{N(\mathfrak{p})^2-\chi(\mathfrak{p})}\right) \\ &= \frac{4}{7}N(m)\beta_{(2)}(m)\beta_{(3)}(m)\left(\prod_{(2),(3)\neq\mathfrak{p}|m}\frac{\beta_{\mathfrak{p}}(m)N(\mathfrak{p})^2}{N(\mathfrak{p})^2-\chi(\mathfrak{p})}\right). \end{aligned}$$

Lemma 4.3.4.

$$\beta_{(3)}(m) = \begin{cases} 8 \cdot \left(\frac{103}{5} \left(\frac{9^{2N}-1}{9^{2(N+1)}} \right) + \frac{1}{9^{2N+1}} \right), & \text{ord}_{(3)}(m) = 2N \\ 8 \cdot \left(\frac{103}{5} \left(\frac{9^{2N}-1}{9^{2(N+1)}} \right) + \frac{182}{9^{2N+3}} \right), & \text{ord}_{(3)}(m) = 2N+1. \end{cases}$$

Proof. All solutions are of Good type when $(3) \nmid (m)$ with

$$\begin{aligned} \beta_{(3)}(m) &= \lim_{v\to\infty} \frac{r_{(3)^v}(m)}{N(3)^{3v}} \\ &= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(3)^v}(m)}{9^{3v}} \\ &= \lim_{v\to\infty} \frac{N(3)^{3(v-1)} r^{\text{Good}}_{(3)}(m)}{9^{3v}} \\ &= \lim_{v\to\infty} \frac{9^{3(v-1)} r^{\text{Good}}_{(3)}(m)}{9^{3v}} \\ &= \frac{r^{\text{Good}}_{(3)}(m)}{3^6} \\ &= \frac{648}{9^3} = \frac{8}{9}. \end{aligned}$$

Now suppose that $\text{ord}_3(m) = 1$, which leads to Bad-Type I and Good solutions. Allowing $Q' = \langle 1, 1, 1, 3+\sqrt{5} \rangle$, we have

$$\begin{aligned} \beta_{(3)}(m) &= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(3)^v}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Bad-Type-I}}_{(3)^v}(m)}{9^{3v}} \\ &= \lim_{v\to\infty} \frac{r^{\text{Good}}_{3^{1+(v-1)}}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{N(3)^2 r^{\text{Good}}_{(3)^{v-1},Q'}(m/3)}{9^{3v}} \\ &= \lim_{v\to\infty} \frac{9^{3(v-1)} r_{(3)}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{9^2 r^{\text{Good}}_{(3)^{1+(v-2)},Q'}(m/3)}{9^{3v}} \\ &= \frac{1376}{9^3} + \frac{9^2 9^{3(v-2)} r^{\text{Good}}_{(3),Q'}(m/3)}{9^{3v}} \\ &= \frac{1376}{9^3} + \frac{r^{\text{Good}}_{(3),Q'}(m/3)}{9^4} \\ &= \frac{1376}{9^3} + \frac{80}{9^3} \\ &= \frac{1456}{9^3}. \end{aligned}$$

When $\mathrm{ord}_3(m) = 2$, Good, Bad-Type I, and Zero solutions occur with

$$\begin{aligned}
\beta_{(3)}(m) &= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(3)^v}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Bad-Type-I}}_{(3)^v}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Zero}}_{(3)^v}(m)}{9^{3v}} \\
&= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(3)^{1+(v-1)}}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{9^2 r^{\text{Good}}_{(3)^{v-1},Q'}(m/3)}{9^{3v}} + \lim_{v\to\infty} \frac{9^4 r_{(3)^{v-2}}(m/9)}{9^{3v}} \\
&= \lim_{v\to\infty} \frac{9^{3(v-1)} r^{\text{Good}}_{(3)}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{9^2 9^{3(v-2)} r^{\text{Good}}_{(3),Q'}(m/3)}{9^{3v}} + \lim_{v\to\infty} \frac{9^4 r^{\text{Good}}_{(3)^{1+(v-3)}}(m/9)}{9^{3v}} \\
&= \frac{1376}{9^3} + \frac{800}{9^4} + \lim_{v\to\infty} \frac{9^4 9^{3(v-3)} r^{\text{Good}}_{(3)}(m/9)}{9^{3v}} \\
&= \frac{1376}{9^3} + \frac{800}{9^4} + \frac{648}{9^5}.
\end{aligned}$$

And in general, we have for $\mathrm{ord}_3(m) = 2N$, $N \in \mathbb{N} \cup \{0\}$:

$$\begin{aligned}
\beta_{(3)}(m) &= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(3)^v}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Bad-Type-I}}_{(3)^v}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Zero}}_{(3)^v}(m)}{9^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{9^{3v}} \left(\sum_{i=0}^{N-1} 9^{4i} r^{\text{Good}}_{(3)^{v-2i}}(m/3^{2i}) \right) + \lim_{v\to\infty} \frac{1}{9^{3v}} \left(\sum_{i=0}^{N-1} 9^{4i} r^{\text{Bad-Type-I}}_{(3)^{v-2i}}(m/3^{2i}) \right) \\
&\quad + \lim_{v\to\infty} \frac{9^{4N} r_{(3)^{v-2N}}(m/3^{2N})}{9^{3v}} \\
&= \frac{1376}{9^3} \left(\sum_{i=0}^{N-1} \frac{1}{9^{2i}} \right) + \frac{800}{9^4} \left(\sum_{i=0}^{N-1} \frac{1}{9^{2i}} \right) + \frac{1}{9^{2N}} \left(\frac{648}{9^3} \right) \\
&= \left(\frac{1376}{9^3} + \frac{800}{9^4} \right) \left(\frac{1}{9^2 - 1} \right) \left(\frac{9^{2N} - 1}{9^{2(N-1)}} \right) + \frac{1}{9^{2N}} \left(\frac{648}{9^3} \right) \\
&= \frac{824}{32805} \left(\frac{9^{2N} - 1}{9^{2(N-1)}} \right) + \frac{1}{9^{2N}} \left(\frac{648}{9^3} \right) \\
&= 8 \left(\frac{103}{5} \left(\frac{9^{2N} - 1}{9^{2(N+1)}} \right) + \frac{1}{9^{2N+1}} \right).
\end{aligned}$$

Similarly, for $\mathrm{ord}_3(m) = 2N+1$, $N \in \mathbb{N} \cup \{0\}$:

$$\begin{aligned}
\beta_{(3)}(m) &= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(3)^v}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Bad-Type-I}}_{(3)^v}(m)}{9^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Zero}}_{(3)^v}(m)}{9^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{9^{3v}} \left(\sum_{i=0}^{N-1} 9^{4i} r^{\text{Good}}_{(3)^{v-2i}}(m/3^{2i}) \right) + \lim_{v\to\infty} \frac{1}{9^{3v}} \left(\sum_{i=0}^{N-1} 9^{4i} r^{\text{Bad-Type-I}}_{(3)^{v-2i}}(m/3^{2i}) \right) \\
&\quad + \lim_{v\to\infty} \frac{9^{4N} r_{(3)^{v-2N}}(m/3^{2N})}{9^{3v}} \\
&= \frac{1376}{9^3} \left(\sum_{i=0}^{N-1} \frac{1}{9^{2i}} \right) + \frac{800}{9^4} \left(\sum_{i=0}^{N-1} \frac{1}{9^{2i}} \right) + \frac{1}{9^{2N}} \left(\frac{1376}{9^3} + \frac{720}{9^4} \right) \\
&= \frac{824}{32805} \left(\frac{9^{2N}-1}{9^{2(N-1)}} \right) + \frac{1}{9^{2N}} \left(\frac{1456}{9^3} \right) \\
&= 8 \left(\frac{103}{5} \left(\frac{9^{2N}-1}{9^{2(N+1)}} \right) + \frac{182}{9^{2N+3}} \right).
\end{aligned}$$

□

Lemma 4.3.5.

$$\beta_2(m) = \begin{cases} 1, & \mathrm{ord}_{(2)}(m) = 2N+1 \\ \dfrac{4^{2N}-1}{4^{2N}} + \dfrac{1}{4^{2N}} & \begin{cases} 15/16, & \mathrm{ord}_{(2)}(m) = 2N, \overline{m/2N} \in C1 \\ 17/16, & \mathrm{ord}_{(2)}(m) = 2N, \overline{m/2N} \in C2 \end{cases} \end{cases}$$

where C1 and C2 are described below.

Proof. Suppose $\mathrm{ord}_2(m) = 2N+1$, $N \in \mathbb{N} \cup \{0\}$. We have:

$$\begin{aligned}
\beta_{(2)}(m) &= \lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{(2)^v}(m)}{4^{3v}} + \lim_{v\to\infty} \frac{r^{\mathrm{Zero}}_{(2)^v}(m)}{4^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{4^{3v}} \left(\sum_{i=0}^{N-1} 4^{4i} r^{\mathrm{Good}}_{(2)^{v-2i}}(m/2^{2i}) \right) + \lim_{v\to\infty} \frac{4^{4N} r_{(2)^{v-2N}}(m/2^{2N})}{4^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{4^{3v}} \left(\sum_{i=0}^{N-1} 4^{4i} 4^{3(v-2i-3)} r^{\mathrm{Good}}_{(8)}(m/2^{2i}) \right) + \lim_{v\to\infty} \frac{4^{4N} 4^{3(v-2N-3)} r_{(8)}(m/2^{2N})}{4^{3v}} \\
&= \sum_{i=0}^{N-1} \frac{r^{\mathrm{Good}}_{(8)}(m/2^{2i})}{4^9 4^{2i}} + \frac{r^{\mathrm{Good}}_{(8)}(m/2^{2N})}{4^9 4^{2N}} \\
&= \sum_{i=0}^{N-1} \frac{245760}{4^9 4^{2i}} + \frac{262144}{4^9 4^{2N}} \\
&= \frac{15}{16} \left(\sum_{i=0}^{N-1} \frac{1}{4^{2i}} \right) + \frac{1}{4^{2N}} \\
&= \frac{4^{2N}-1}{4^{2N}} + \frac{1}{4^{2N}} \\
&= 1.
\end{aligned}$$

Let $\theta = (1+\sqrt{5})/2$. When m is odd, we have two categories for representatives of $m \pmod 8$:

Table 4.2: $C1$, $C2$ values

C1	C1	C1	C2	C2	C2
$1+3\theta$	$1+4\theta$	$1+5\theta$	θ	3θ	5θ
$1+6\theta$	$2+\theta$	$2+5\theta$	7θ	1	$1+\theta$
$3+\theta$	$3+2\theta$	$3+4\theta$	$1+2\theta$	$1+7\theta$	$2+3\theta$
$3+7\theta$	$4+\theta$	$4+3\theta$	$2+7\theta$	3	$3+3\theta$
$4+5\theta$	$4+7\theta$	$5+\theta$	$3+5\theta$	$3+6\theta$	5
$5+4\theta$	$5+6\theta$	$5+7\theta$	$5+2\theta$	$5+3\theta$	$5+5\theta$
$6+3\theta$	$6+7\theta$	$7+2\theta$	$6+\theta$	$6+5\theta$	7
$7+3\theta$	$7+4\theta$	$7+5\theta$	$7+\theta$	$7+6\theta$	$7+7\theta$

When $(2) \nmid (m)$, all solutions are of Good type and

$$\begin{aligned}
\beta_{(2)}(m) &= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(2)^v}(m)}{4^{3v}} \\
&= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(2)^{3+(v-3)}}(m)}{4^{3v}} \\
&= \lim_{v\to\infty} \frac{4^{3(v-3)} r^{\text{Good}}_{(8)}(m)}{4^{3v}} \\
&= \frac{r^{\text{Good}}_{8}(m)}{4^9} \\
&= \begin{cases} 15/2^4, & \overline{m} \in \text{C1} \\ 17/2^4, & \overline{m} \in \text{C2}. \end{cases}
\end{aligned}$$

Last, we summarize the case $\text{ord}_2(m) = 2N$, $N \in \mathbb{N} \cup \{0\}$.

$$\begin{aligned}
\beta_{(2)}(m) &= \lim_{v\to\infty} \frac{r_{(2)^v}(m)}{4^{3v}} \\
&= \lim_{v\to\infty} \frac{r^{\text{Good}}_{(2)^v}(m)}{4^{3v}} + \lim_{v\to\infty} \frac{r^{\text{Zero}}_{(2)^v}(m)}{4^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{4^{3v}} \left(\sum_{i=0}^{N-1} 4^{4i} r^{\text{Good}}_{(2)^{v-2i}}(m/2^{2i}) 4^{3v} \right) + \lim_{v\to\infty} \frac{4^{4N} r_{(2)^{v-2N}}(m/2^{2N})}{4^{3v}} \\
&= \lim_{v\to\infty} \frac{1}{4^{3v}} \left(\sum_{i=0}^{N-1} 4^{4i} 4^{3(v-2i-3)} r^{\text{Good}}_{(8)}(m/2^{2i}) \right) + \lim_{v\to\infty} \frac{4^{4N} 4^{3(v-2N-3)} r_{(8)}(m/2^{2N})}{4^{3v}} \\
&= \sum_{i=0}^{N-1} \frac{r^{\text{Good}}_{8}(m/2^{2i})}{4^9 4^{2i}} + \frac{r^{\text{Good}}_{8}(m/2^{2N})}{4^9 4^{2N}} \\
&= \sum_{i=0}^{N-1} \frac{245760}{4^9 4^{2i}} + \frac{1}{4^9 4^{2N}} \begin{cases} 245760, & \overline{m/2N} \in \text{C1} \\ 278528, & \overline{m/2N} \in \text{C2} \end{cases} \\
&= \frac{15}{16} \left(\sum_{i=0}^{N-1} \frac{1}{4^{2i}} \right) + \frac{1}{4^{2N}} \begin{cases} 15/16, & \overline{m/2N} \in \text{C1} \\ 17/16, & \overline{m/2N} \in \text{C2} \end{cases} \\
&= \frac{4^{2N}-1}{4^{2N}} + \frac{1}{4^{2N}} \begin{cases} 15/16, & \overline{m/2N} \in \text{C1} \\ 17/16, & \overline{m/2N} \in \text{C2}. \end{cases}
\end{aligned}$$

□

Lemma 4.3.6. *Let* $(2), (3) \neq \mathfrak{p}$ *be a prime which divides* (m).

$$\beta_{\mathfrak{p}}(m) \cdot \frac{N(\mathfrak{p})^2}{N(\mathfrak{p})^2 - \chi(\mathfrak{p})} = \begin{cases} \frac{1}{N(\mathfrak{p})^{2N+1}} \left(\frac{N(\mathfrak{p})^{2N+2} - 1}{N(\mathfrak{p}) - 1} \right), & \mathrm{ord}_{\mathfrak{p}}(m) = 2N+1, \chi(\mathfrak{p}) = 1 \\ \frac{1}{N(\mathfrak{p})^{2N+1}} \left(\frac{N(\mathfrak{p})^{2N+2} - 1}{N(\mathfrak{p}) + 1} \right), & \mathrm{ord}_{\mathfrak{p}}(m) = 2N+1, \chi(\mathfrak{p}) = -1 \\ \frac{1}{N(\mathfrak{p})^{2N}} \left(\frac{N(\mathfrak{p})^{2N+1} - 1}{N(\mathfrak{p}) - 1} \right), & \mathrm{ord}_{\mathfrak{p}}(m) = 2N, \chi(\mathfrak{p}) = 1 \\ \frac{1}{N(\mathfrak{p})^{2N}} \left(\frac{N(\mathfrak{p})^{2N+1} + 1}{N(\mathfrak{p}) + 1} \right), & \mathrm{ord}_{\mathfrak{p}}(m) = 2N, \chi(\mathfrak{p}) = -1. \end{cases}$$

Proof. Suppose $\mathrm{ord}_{\mathfrak{p}}(m) = 2N$, $N \in \mathbb{N} \cup \{0\}$. Then all solutions are Zero or Good type with

$$\begin{aligned} \beta_{\mathfrak{p}}(m) &= \lim_{v \to \infty} \frac{r_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}} \\ &= \left(\lim_{v \to \infty} \frac{r^{\mathrm{Good}}_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}} \right) + \left(\lim_{v \to \infty} \frac{r^{\mathrm{Zero}}_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}} \right) \\ &= \frac{N(\mathfrak{p})^3 + N(\mathfrak{p})\left(N(\mathfrak{p}) - 1\right)\chi(\mathfrak{p}) - 1}{N(\mathfrak{p})^3} \\ &\quad + \lim_{v \to \infty} \frac{1}{N(\mathfrak{p})^{3v}} \left(\left(\sum_{i=1}^{N-1} N(\mathfrak{p})^{4i} r^{\mathrm{Good}}_{\mathfrak{p}^{v-2i}}(m/\pi_{\mathfrak{p}}^{2i}) \right) + N(\mathfrak{p})^{4N} r_{\mathfrak{p}^{v-2N}}(m/\pi_{\mathfrak{p}}^{2N}) \right) \\ &= \left(\sum_{i=0}^{N-1} N(\mathfrak{p})^{-2i} \right) \cdot \left(\frac{N(\mathfrak{p})^3 + N(\mathfrak{p})\left(N(\mathfrak{p}) - 1\right)\chi(\mathfrak{p}) - 1}{N(\mathfrak{p})^3} \right) + N(\mathfrak{p})^{-2N} \left(\frac{N(\mathfrak{p})^2 - \chi(\mathfrak{p})}{N(\mathfrak{p})^2} \right). \end{aligned}$$

Therefore,

$$\begin{aligned} \beta_{\mathfrak{p}}(m) \cdot \frac{N(\mathfrak{p})^2}{N(\mathfrak{p})^2 - \chi(\mathfrak{p})} &= \begin{cases} \sum_{i=0}^{2N} N(\mathfrak{p})^{-i}, & \chi(\mathfrak{p}) = 1 \\ \sum_{i=0}^{2N} (-1)^i N(\mathfrak{p})^{-i}, & \chi(\mathfrak{p}) = -1 \end{cases} \\ &= \begin{cases} \frac{1}{N(\mathfrak{p})^{2N}} \left(\frac{N(\mathfrak{p})^{2N+1} - 1}{N(\mathfrak{p}) - 1} \right), & \chi(\mathfrak{p}) = 1 \\ \frac{1}{N(\mathfrak{p})^{2N}} \left(\frac{N(\mathfrak{p})^{2N+1} + 1}{N(\mathfrak{p}) + 1} \right), & \chi(\mathfrak{p}) = -1. \end{cases} \end{aligned}$$

Last, suppose $\mathrm{ord}_{\mathfrak{p}}(m) = 2N+1$, $N \in \mathbb{N} \cup \{0\}$. Again, all solutions are of Zero or Good type and

$$\begin{aligned}
\beta_{\mathfrak{p}}(m) &= \lim_{v\to\infty} \frac{r_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}} \\
&= \left(\lim_{v\to\infty} \frac{r^{\mathrm{Good}}_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}}\right) + \left(\lim_{v\to\infty} \frac{r^{\mathrm{Zero}}_{\mathfrak{p}^v}(m)}{N(\mathfrak{p})^{3v}}\right) \\
&= \frac{N(\mathfrak{p})^3 + N(\mathfrak{p})\left(N(\mathfrak{p})-1\right)\chi(\mathfrak{p}) - 1}{N(\mathfrak{p})^3} \\
&\quad + \lim_{v\to\infty} \frac{1}{N(\mathfrak{p})^{3v}} \left(\sum_{j=1}^{N} N(\mathfrak{p})^{4j} N(\mathfrak{p})^{3(v-2j-1)} \left(N(\mathfrak{p})^3 + N(\mathfrak{p})\left(N(\mathfrak{p})-1\right)\chi(\mathfrak{p}) - 1\right)\right) \\
&= \frac{N(\mathfrak{p})^3 + N(\mathfrak{p})\left(N(\mathfrak{p})-1\right)\chi(\mathfrak{p}) - 1}{N(\mathfrak{p})^3} + \sum_{j=1}^{N} N(\mathfrak{p})^{-2j} \left(\frac{N(\mathfrak{p})^3 + N(\mathfrak{p})\left(N(\mathfrak{p})-1\right)\chi(\mathfrak{p}) - 1}{N(\mathfrak{p})^3}\right) \\
&= \sum_{j=0}^{N} N(\mathfrak{p})^{-2j} \left(\frac{N(\mathfrak{p})^3 + N(\mathfrak{p})\left(N(\mathfrak{p})-1\right)\chi(\mathfrak{p}) - 1}{N(\mathfrak{p})^3}\right).
\end{aligned}$$

Thus

$$\begin{aligned}
\beta_{\mathfrak{p}}(m) \cdot \frac{N(\mathfrak{p})^2}{N(\mathfrak{p})^2 - \chi(\mathfrak{p})} &= \begin{cases} \displaystyle\sum_{j=0}^{2N+1} N(\mathfrak{p})^{-j}, & \chi(\mathfrak{p}) = 1 \\ \displaystyle\sum_{j=0}^{2N+1} (-1)^j N(\mathfrak{p})^{-j}, & \chi(\mathfrak{p}) = -1 \end{cases} \\
&= \begin{cases} \dfrac{1}{N(\mathfrak{p})^{2N+1}} \left(\dfrac{N(\mathfrak{p})^{2N+2} - 1}{N(\mathfrak{p}) - 1}\right), & \chi(\mathfrak{p}) = 1 \\ \dfrac{1}{N(\mathfrak{p})^{2N+1}} \left(\dfrac{N(\mathfrak{p})^{2N+2} - 1}{N(\mathfrak{p}) + 1}\right), & \chi(\mathfrak{p}) = -1. \end{cases}
\end{aligned}$$

This series of lemmas completely determines $a_E(m)$ for any $m \in \mathcal{O}_K^+$. □

Chapter 5

Bibliography

[1] Andrianov, A.N., Zhuravlev, V.G., *Modular Forms and Hecke Operators*, Translated from the Russian, American Mathematical Society, 1995.

[2] Apostol, T., *Introduction to Analytic Number Theory*, Springer-Verlag, New York, Heidelberg, Berlin 1976.

[3] Bhargava, M., Hanke., J, *Universal Quadratic Forms and the 290 Theorem*, preprint

[4] Bump, D., *Automorphic Forms and Representations*, Cambridge University Press, 1998.

[5] Bump, D., etal., *An Introduction to the Langlands Program*, Birkhäuser, Boston, 2004.

[6] Cassels, J.W.S., *Rational Quadratic Forms*, Dover Publications, 2008.

[7] Chan, W-K., Kim, M-H., Raghavan, S., *Ternary Universal Integral Quadratic Forms Over Real Number Fields*, Japanese Journal of Mathm., 22 (1996) no.2., 263-273.

[8] Conway, J.H., Sloane, N.J.A., *Sphere Packings, Lattices, and Groups*, Springer, 1999.

[9] Cox, D.A., *Primes of the form* $x^2 + ny^2$, A Wiley-Interscience Publication, John Wiley & Sons Inc., New York, 1989

[10] Dembélé, L., *Explicit Computations of Hilbert Modular Forms on* $\mathbb{Q}(\sqrt{5})$, Experimental Mathematics, Vol. 14, No 4 (2005), 457-66.

[11] Dembélé, L., *Quaternionic Manin Symbols, Brandt Matrices and Hilbert Modular Forms*, Mathematics of Computations, Vol. 76, No 258 (2007), 1039–1057.

[12] Dembélé, L., Voight, J., *Explicit Methods for Hilbert Modular Forms*, Elliptic curves, Hilbert modular forms and Galois deformations, Birkhauser, Basel, 2013, 135-198.

[13] Deutsch, J., *Geometry of Numbers Proof of Götzky's Four-Squares Theorem*, Journal of Number Theory 96, 2002.

[14] Diamond, F., and Shurman, J., *A First Course in Modular Forms*, Springer, 2005

[15] Freitag, E., *Hilbert Modular Forms*, Springer-Verlag Berlin Heidelberg, 1990.

[16] Garrett,P., *Holomorphic Hilbert Modular Forms*, Wadsworth Inc., Belmont, California,1990.

[17] van der Geer,G., *Hilbert Modular Surfaces*, Springer-Verlag, Berlin, 1988.

[18] Goren, E., *Lectures on Hilbert Modular Varieties and Modular Forms*, American Mathematical Society, 2002.

[19] Götzky, F., *Über eine zahlentheoretische Anwendung von Modulfunktionen zweier Veränderlicher*, Mathematische Annalen, 1928.

[20] Greenberg, M., Voight, J., *Computing Systems of Hecke Eigenvalues Associated to Hilbert Modular Forms*, Mathematics of Computation, Volume 80, Number 274, 2011.

[21] Hanke, J., *Local Densities and Explicit Bounds for Representability by a Quadratic Form*, Duke Journal of Mathematics, 2004

[22] Hida, H., *Hilbert Modular Forms and Iwasawa Theory*, Oxford Science Publications, 2006.

[23] Iwasawa, K., *Lectures on p-Adic L-Functions*, Princeton University Press, 1972.

[24] Jacobson, N., *Basic Algebra, Volume I*, W.H. Freeman and Company, 1980.

[25] Johansson, S., *A Description of Quaternion Algebras*, `http://math.chalmers.se/~sj/forskning/structure.ps`.

[26] Kitaoka, Y., *Arithmetic of Quadratic Forms*, Cambridge University Press, 1993.

[27] Knapp, A., *Elliptic Curves*, Princeton University Press, 1992.

[28] Lam, T.Y., *Introduction to Quadratic Forms Over Fields*, American Mathematical Society, Graduate Studeis in Mathematics Volume 67, Providence, RI, 2005.

[29] MAGMA

[30] Miyake, T., *Modular Forms*, Springer, 1989.

[31] Neukirch, J., *Algebraic Number Theory*, Springer-Verlag, Berlin, Heidelberg, 1999.

[32] O'Meara, T.O., *Introduction to Quadratic Forms*, Springer-Verlag, Berlin, Heidelberg, 1963.

[33] Pizer, A., *An Algorithm for Computing Modular Forms on* $\Gamma_0(N)$, Journal of Algebra 64, 340-390 (1980).

[34] Serre, J.P., *A Course in Arithmetic*, Springer-Verlag, New York, 1973, Translated from the French, Graduate Texts in Mathematics, No. 7.

[35] Shimura, G., *Introduction to the Arithmetic Theory of Automorphic Functions*, Publications of the Mathematical Society of Japan, No. 11. Iwanami Shoten, Publishers, Tokyo, 1971.

[36] Shimura, G., *Arithmetic of Quadratic Forms*, Springer Monographs in Mathematics, 2010.

[37] Shimura, G., *The Special Values of the Zeta Functions Associated With Hilbert Modular Forms*, Duke Mathematical Journal, September 1978.

[38] Siegel, C. L., *Über Die Analytische Theorie Der Quadratischen Formen III*, Annals of Mathematics, 1937.

[39] Stein, W.A., *Modular Forms, a Computational Approach*, American Mathematical Society, Graduate Studies in Mathematics Volume 79, Providence, RI, 2007.

[40] W.A. Stein et al., *Sage Mathematics Software (Version 5.6)*, The Sage Development Team, 2013, `http://www.sagemath.org`

[41] Vignéras,M.-F., *Arithmetique des algebres de quaternions*, Lecture Notes in Mathematics, vol. 800, Springer, Berlin, 1980.

[42] Voight, J., *Identifying the Matrix Ring: Algorithms for Quaternion Algebras and Quadratic Forms*, Quadratic and higher degree forms, Developments in Math., vol. 31, Springer, New York, 2013, 255-298.

Appendix A

Appendices

A.1 Code for $r_Q(m)$

Below is the Sage code introduced in the Bounding Regions chapter.

```
#####################################################
#This code will create bounding boxes
#for quaternary, totally positive,
#quadratic forms over totally real number fields
#that can be used to calculate the number of
#representations of a totally positive integer
#by the form.
#####################################################

def change_of_basis_return(d,Q):

    """
    This will take a quadratic form in four variables, over
    K, diagonalize it over K, and then return the INVERSE of
    that matrix.

    EXAMPLES:
    =====
    sage: x = var('x')
    sage: K.<a>=NumberField(x^2-5,embedding=1)
    sage: Q=Matrix(K,4,[1,0,0,0,0,1,0,0,0,0,1,4,0,0,0,5])
```

```
sage: change_of_basis_return(5,Q)
[1 0 0 0]
[0 1 0 0]
[0 0 1 2]
[0 0 0 1]
"""

assert d.is_squarefree() == True
assert d>0

x = QQ['x'].0
K.<a> = NumberField(x^2-d,embedding=1)
OK = K.ring_of_integers()



assert Q.matrix_space() == MatrixSpace(K,4,4)

QasQF = QuadraticForm(K,(Q+Q.transpose())/2)
R = QasQF.rational_diagonal_form(return_matrix=True)
Mat = R[1]
```

```
    return Mat.inverse()



def is_totally_positive_definite_quaternary(d,Q):
    """
    This checks whether or not a given 4X4 matrix (either symmetric or upper triangular), representing
    a quadratic form over Q(sqrt(d)), d>0 is totally positive definite.
    The method is diagonalizing the form over
    the field and then checking whether or not the diagonal coefficients are totally positive.
    =========
    EXAMPLES:
    sage: x = var('x')
    sage: K.<a>=NumberField(x^2-5,embedding=1)
    sage: Q=Matrix(K,4,[-1,0,0,0,0,1,0,0,0,0,1,2,0,0,2,5])
    sage: Q
    [   -1     0     0     0]
    [    0     1     0     0]
    [    0     0     1     2]
    [    0     0     2     5]
```

```
sage: is_totally_positive_definite_quaternary(5,Q)
False

=========
sage: x = var('x')
sage: K.<a>=NumberField(x^2-2,embedding=1)
sage: Q=Matrix(K,4,[2+a,0,0,0,0,1,0,0,0,0,1,4,0,0,0,5])
sage: Q
[a + 2     0     0     0]
[    0     1     0     0]
[    0     0     1     4]
[    0     0     0     5]
sage: is_totally_positive_definite_quaternary(2,Q)
True

"""
assert d.is_squarefree() == True
assert d>0

x = QQ['x'].0
K.<a> = NumberField(x^2-d,embedding=1)
```

```
    OK = K.ring_of_integers()

    assert Q.matrix_space() == MatrixSpace(K,4,4)

    QasQF = QuadraticForm(K,(Q+Q.transpose())/2)
    R = QasQF.rational_diagonal_form()
    GRAM = R.Gram_matrix()

    if GRAM[0][0].is_totally_positive() == True and GRAM[1][1].is_totally_positive() == True
    and GRAM[2][2].is_totally_positive() == True and GRAM[3][3].is_totally_positive() == True:
        return True
    else:
        return False




def integer_coefficients_of_representation(d,A):
    """
    Given K=QQ(sqrt(d)) a real quadratic number field, and an element A of O_K,
    A=x+y(theta) where theta=sqrt(d) (if d is 2,3 mod 4) or theta=(1+sqrt(d))/2 (if d is 1 mod 4).
```

```
This function returns (x,y).

EXAMPLES:
==========

sage: x = var('x')
sage: K.<a>=NumberField(x^2-2,embedding=1)
sage: A=(2+a)
sage: integer_coefficients_of_representation(2,A)
(2, 1)

=========
sage: x = var('x')
sage: K.<a>=NumberField(x^2-2,embedding=1)
sage: A=(2+a)
sage: integer_coefficients_of_representation(5,A)
AssertionError

========
sage: x = var('x')
sage: K.<a>=NumberField(x^2-5,embedding=1)
```

```
sage: integer_coefficients_of_representation(5,a)
(-1, 2)
"""
assert d.is_squarefree() == True
assert d>0

x = QQ['x'].0
K.<a> = NumberField(x^2-d, embedding=1)

#assert A.parent() == K --needed or else actual integers won't feed through!
assert A.is_integral() == True

if d%4 ==1:
    y = ((2*K(A))-K(A).trace())/a
    x = (K(A).trace() - y)/2
    return (x,y)
else:
    x = (K(A).trace())/2
    y = (K(A)-x)/a
    return (x,y)
```

```
def integer_coefficients_of_conjugate_representation(d,A):
    """
    This code inputs d, for a totally real quadratic number field QQ(sqrt(d)),
    and an integer A=x+y(theta), where theta = sqrt(d) (if d is congruent to 2 or 3 mod 4)
    and theta = (1+sqrt(d))/2 (if d is congruent to 1 mod 4). Given these two inputs, the code
    returns the conjugate of A, A'=x'+y'.
    """
    assert d.is_squarefree() == True
    assert d>0

    x = QQ['x'].0
    K.<a> = NumberField(x^2-d, embedding=1)

    assert A.is_integral() == True

    if d%4 ==1:
        y = ((2*K(A))-K(A).trace())/a
        x = (K(A).trace() - y)/2
        return (x+y,-y)
    else:
        x = (K(A).trace())/2
```

```
        y = (K(A)-x)/a
        return (x,-y)


def find_D(d):
    """
    This program merely churns out one of two constants, depending upon d's congruence mod 4.

    EXAMPLES:
    =========
    sage: find_D(-2)
    Traceback (most recent call last):
    ...
    AssertionError

    ==========
    sage: find_D(17)
    1/8

    ===========
    sage: find_D(88)
```

```
    Traceback (most recent call last):
    ...
    AssertionError


    ===========
    sage: find_D(46)
    1/92
    """
    assert d.is_squarefree() == True
    assert d>0

    if d%4==1:
        return 2/(d-1)
    else:
        return 1/(2*d)



def List_of_Totally_Positive_with_Fixed_Trace(d,n):
    """
    This code considers a real quadratic number field Q(sqrt(d)), and
```

```
a positive integer n, and computes all totally positive integers in Q(sqrt(d))
with trace n.

EXAMPLES:
==========
sage: List_of_Totally_Positive_with_Fixed_Trace(5,14)
[(4, 6),(5, 4),(6, 2),(7, 0),(8, -2),(9, -4),(10, -6)]

==========
sage: List_of_Totally_Positive_with_Fixed_Trace(3,1)
Not an acceptable value!

===========
sage: List_of_Totally_Positive_with_Fixed_Trace(22,2)
[(1, 0)]

============
sage: List_of_Totally_Positive_with_Fixed_Trace(13,5)
[(2, 1),(3, -1)]

============
```

```
sage: List_of_Totally_Positive_with_Fixed_Trace(17,-5)
Traceback (most recent call last):
...
AssertionError
"""
assert d.is_squarefree() == True
assert d>0
x = QQ['x'].0
K.<a> = NumberField(x^2-d, embedding=1)
L=[]


if d%4 ==1:
    assert n>=0
    for i in range(n*(sqrt(d)-1)/(2*sqrt(d)),n*(1+sqrt(d))/(2* sqrt(d))+1):
        if (i+(n-2*i)*((1+a)/2)).is_totally_positive() == True:
            L.append((i, n-2*i))
    return L
else:


    assert n>=0
    if n%2 ==0:
```

```
            i = n/2
            for j in range(-(i/sqrt(d)), (i/sqrt(d))+1):
                if (i+j*a).is_totally_positive() == True:
                    L.append((i,j))
            return L
        else:
                print 'Not an acceptable value!'
```



```
def Dictionary_According_to_Trace(d,n):
    """
    This code considers a real quadratic number field Q(sqrt(d)), and
    a positive integer n, and computes all totally positive integers in Q(sqrt(d))
    with trace less-than-or-equal-to n.

    EXAMPLES:
    =========
```

```
sage: Dictionary_According_to_Trace(5,2)
[(1, 0)]

=========
sage: Dictionary_According_to_Trace(2,6)
[(1, 0), (2, -1), (2, 0), (2, 1), (3, -2), (3, -1), (3, 0), (3, 1), (3, 2)]

=========
sage: Dictionary_According_to_Trace(19,3)
[(1, 0)]
"""
assert d.is_squarefree() == True
assert d>0
x = QQ['x'].0
K.<a> = NumberField(x^2-d, embedding=1)
Dictionary=[]
if d%4 ==1:
    assert n>0
    for k in range(n+1):
        for l in range(len(List_of_Totally_Positive_with_Fixed_Trace(d,k))):
            Dictionary.append(List_of_Totally_Positive_with_Fixed_Trace(d,k)[l])
```

```
            return Dictionary
        else:
            for k in range(n+1):
                if k%2 ==0:
                    for l in range(len(List_of_Totally_Positive_with_Fixed_Trace(d,k))):
                        Dictionary.append(List_of_Totally_Positive_with_Fixed_Trace(d,k)[l])
            return Dictionary




def Conjugate(d,m):
    """
    This returns the conjugate of m, which is an element
    of the number field QQ(sqrt(d)).
    """
    assert d.is_squarefree() == True
    assert d>0
    x = QQ['x'].0
    K.<a> = NumberField(x^2-d, embedding=1)

    #assert m.parent() == K
```

```
    return m.trace()-m


def List_for_all_Theta(d,Q,m):
    """
    As described in sumofsquares.pdf, when defining a bounded region associated to r_Q(m),
    the regions associated to r_Q(n) are properly contained, where n depends on Q and m.
    This command lists all of the n.

    EXAMPLES:
    ========
    x = var('x')
    K.<a>=NumberField(x^2-5,embedding=1)
    R=Matrix(K,4,[1,0,0,0,  0,1,0,0,  0,0,2,0,  0,0,0,8])

    List_for_all_Theta(5,R,8)
    [(1, 0), (1, 1), (2, -1), (2, 0), (2, 1), (2, 2), (2, 3), (3, -1), (3,
    0), (3, 1), (3, 2), (3, 3), (3, 4), (4, -2), (4, -1), (4, 0), (4, 1),
    (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (5, -3), (5, -2), (5, -1), (5,
    0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, -3), (6, -2),
```

```
(6, -1), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (7, -4), (7, -3), (7,
-2), (7, -1), (7, 0), (7, 1), (7, 2), (8, -4), (8, -3), (8, -2), (8,
-1), (8, 0), (9, -5), (9, -4), (9, -3), (9, -2), (10, -6), (10, -5),
(10, -4), (11, -6)]



========
x = var('x')
K.<a>=NumberField(x^2-22,embedding=1)
R=Matrix(K,4,[1,1,1,0,  0,1,1,1,  0,0,1,1,  0,0,0,3])


List_for_all_Theta(22,R,5)
[(1, 0), (2, 0), (3, 0), (4, 0), (5, -1), (5, 0), (5, 1)]


"""
assert d.is_squarefree() == True
assert d>0
assert is_totally_positive_definite_quaternary(d,Q) == True


x = QQ['x'].0
K.<a> = NumberField(x^2-d,embedding=1)
```

```
OK = K.ring_of_integers()


QasQF = QuadraticForm(K, (Q+Q.transpose())/2)
FQ = QasQF.polynomial(names='z')
Diag = (QasQF.rational_diagonal_form()).Gram_matrix()
D = 2*Diag




B3 = Conjugate(d,K(D[3][3]))*(K(m/D[3][3]).trace())
M3 = Conjugate(d, K(D[3][3]))/ (D[3][3])

B2 = Conjugate(d,K(D[2][2]))*(K(m/D[2][2]).trace())
M2 = Conjugate(d, K(D[2][2]))/ (D[2][2])

B1 = Conjugate(d, K(D[1][1]))*(K(m/D[1][1]).trace())
M1 = Conjugate(d, K(D[1][1]))/ (D[1][1])

B0 = Conjugate(d, K(D[0][0]))*(K(m/D[0][0]).trace())
M0 = Conjugate(d, D[0][0])/ (D[0][0])
```

```
if d%4==1:
    T=(1+a)/2
    TB = Conjugate(d, T)
    IBT = 1/((1/T)-(1/TB))
    N3=[]
    N2=[]
    N1=[]
    N0=[]
    N=[]
    for i3 in range(1,RR(B3*IBT*(1/(M3*T)-1/TB)) +1):
        for j3 in range(floor(-RR(B3/a)),RR(B3/(M3*a))+1):
            if RR(Conjugate(d, i3+j3*T)) <= RR(B3 - M3*(i3+j3*T)):
                N3.append((i3,j3))
    for i2 in range(1, RR(B2*IBT*(1/(M2*T)-1/TB)) +1):
        for j2 in range(floor(-RR(B2/a)), RR(B2/(M2*a)) +1):
            if RR(Conjugate(d, i2+j2*T)) <= RR(B2-M2*(i2+j2*T)):
                N2.append((i2,j2))
    for i1 in range(1, RR(B1*IBT*(1/(M1*T)-1/TB))+1):
        for j1 in range(floor(-RR(B1/a)),RR(B1/(M1*a)) +1):
            if RR(Conjugate(d, i1+j1*T)) <= RR(B1-M1*(i1+j1*T)):
                N1.append((i1,j1))
```

```
    for i0 in range(1, RR(B0*IBT*(1/(M0*T)-1/TB)) +1):
        for j0 in range(floor(-RR(B0/a)), RR(B0/(M0*a)) +1):
            if RR(Conjugate(d, i0+j0*T)) <= RR(B0-M0*(i0+j0*T)):
                N0.append((i0,j0))
    for k in N3:
        if N2.count(k)>0:
            if N1.count(k)>0:
                if N0.count(k)>0:
                    if K(k[0]+k[1]*T).is_totally_positive() == True:
                        N.append(k)
    return N
else:
    N3=[]
    N2=[]
    N1=[]
    N0=[]
    N=[]
    for i3 in range(1,RR((B3/2)*(1+M3))+1):
        for j3 in range(floor(RR(-B3/(2*a))), RR(B3/(M3*2*a))+1):
            if RR(Conjugate(d, i3+j3*a)) <=RR(B3-M3*(i3+j3*a)):
                N3.append((i3,j3))
```

```
for i2 in range(1,RR((B2/2)*(1+M2))+1):
    for j2 in range(floor(RR(-B2/(2*a))), RR(B2/(M2*2*a))+1):
        if RR(Conjugate(d, i2+j2*a)) <=RR(B2-M2*(i2+j2*a)):
            N2.append((i2,j2))
for i1 in range(1,RR((B1/2)*(1+M1))+1):
    for j1 in range(floor(RR(-B1/(2*a))), RR(B1/(M1*2*a))+1):
        if RR(Conjugate(d, i1+j1*a)) <=RR(B1-M1*(i1+j1*a)):
            N1.append((i1,j1))
for i0 in range(1,RR((B0/2)*(1+M0))+1):
    for j0 in range(floor(RR(-B0/(2*a))), RR(B1/(M0*2*a)+1)):
        if RR(Conjugate(d, i0+j0*a)) <=RR(B0-M0*(i0+j0*a)):
            N0.append((i0,j0))
for k in N3:
    if N2.count(k)>0:
        if N1.count(k)>0:
            if N0.count(k)>0:
                if K(k[0]+k[1]*(a)).is_totally_positive() == True:
                    N.append(k)
return N
```

```
def Theta_Computation(d,Q,m):
    """
    For every n obtained from List_for_all_Theta(d,Q,m), r_Q(n) is calculated.

    EXAMPLES:
    ========
    x = var('x')
    K.<a>=NumberField(x^2-5,embedding=1)
    R=Matrix(K,4,[1,0,0,0,  0,1,0,0,  0,0,2,0,  0,0,0,8])

    List_for_all_Theta(5,R,8)
    [(1, 0), (1, 1), (2, -1), (2, 0), (2, 1), (2, 2), (2, 3), (3, -1), (3,
    0), (3, 1), (3, 2), (3, 3), (3, 4), (4, -2), (4, -1), (4, 0), (4, 1),
    (4, 2), (4, 3), (4, 4), (4, 5), (4, 6), (5, -3), (5, -2), (5, -1), (5,
    0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5), (5, 6), (6, -3), (6, -2),
    (6, -1), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (7, -4), (7, -3), (7,
    -2), (7, -1), (7, 0), (7, 1), (7, 2), (8, -4), (8, -3), (8, -2), (8,
    -1), (8, 0), (9, -5), (9, -4), (9, -3), (9, -2), (10, -6), (10, -5),
    (10, -4), (11, -6)]
```

```
Theta_Computation(5,R,8)
[4, 4, 4, 6, 8, 6, 4, 8, 16, 8, 8, 16, 8, 6, 8, 12, 24, 24, 24, 12, 8,
6, 4, 8, 24, 28, 24, 16, 16, 24, 28, 24, 16, 24, 24, 32, 16, 24, 16, 24,
8, 24, 16, 16, 32, 32, 40, 12, 16, 24, 32, 56, 8, 24, 16, 40, 6, 28, 24,
24]

========
x = var('x')
K.<a>=NumberField(x^2-22,embedding=1)
R=Matrix(K,4,[1,1,1,0,  0,1,1,1,  0,0,1,1,  0,0,0,3])

List_for_all_Theta(22,R,5)
[(1, 0), (2, 0), (3, 0), (4, 0), (5, -1), (5, 0), (5, 1)]

Theta_Computation(22,R,5)
[12, 6, 36, 28, 0, 72, 0]



========
```

```
x = var('x')
K.<a>=NumberField(x^2-101,embedding=1)
R=Matrix(K,4,[1,1,1,0,  0,2,2,-1,  0,0,3,2,  0,0,0,13])

List_for_all_Theta(101,R,15)
[(1, 0), (2, 0), (3, 0), (4, 0), (5, 0), (5, 1), (6, -1), (6, 0), (6,
1), (7, -1), (7, 0), (7, 1), (8, -1), (8, 0), (8, 1), (9, -1), (9, 0),
(9, 1), (10, -1), (10, 0), (10, 1), (10, 2), (11, -1), (11, 0), (11, 1),
(11, 2), (12, -2), (12, -1), (12, 0), (12, 1), (12, 2), (13, -2), (13,
-1), (13, 0), (13, 1), (13, 2), (14, -2), (14, -1), (14, 0), (14, 1),
(14, 2), (15, -2), (15, -1), (15, 0), (16, -2)]

Theta_Computation(101,R,15)
[2, 4, 6, 10, 4, 0, 0, 2, 0, 0, 14, 0, 0, 8, 0, 0, 10, 0, 0, 8, 0, 0, 0,
6, 0, 0, 0, 0, 14, 0, 0, 0, 0, 16, 0, 0, 0, 0, 22, 0, 0, 0, 0, 20, 0]
"""

assert d.is_squarefree() == True
assert d > 0

assert is_totally_positive_definite_quaternary(d,Q) == True
```

```
x = QQ['x'].0

K.<a> = NumberField(x^2-d,embedding=1)

OK = K.ring_of_integers()



QasQF = QuadraticForm(K,(Q+Q.transpose())/2)

FQ = QasQF.polynomial(names='z')

Diag = (QasQF.rational_diagonal_form()).Gram_matrix()

D = 2*Diag

MM = change_of_basis_return(d,Q)

n = RR(K(m).trace())

List = List_for_all_Theta(d,Q,m)



B3 = K(m/D[3][3]).trace()

B2 = K(m/D[2][2]).trace()

B1 = K(m/D[1][1]).trace()

B0 = K(m/D[0][0]).trace()
```

```
DD = find_D(d)
counter = 0
Counting = [0 for k in range(len(List))]
if d%4 ==1:


    T=(1+a)/2
    TB = Conjugate(d, T)
    IBT = 1/((1/T)-(1/TB))
    for i3 in range(floor(-sqrt(RR(B3))), sqrt(RR(B3)) +1):
        for j3 in range(floor(-sqrt(RR(B3*DD))), sqrt(RR(B3*DD))+1):
            if RR(K(D[3][3]*(i3+j3*T)^2).trace()) <= n:
                for i2 in range(floor(RR(-sqrt(RR(B2))
                +(1/a)*((TB)*MM[2][3]*(i3+j3*T)-Conjugate(d,(TB)*MM[2][3]*
                                (i3+j3*T))))),sqrt(RR(B2))+(1/a)*((TB)*MM[2][3]*(i3+j3*T)
                                -Conjugate(d,(TB)*MM[2][3]*
                                (i3+j3*T))) +1):
                    for j2 in range(floor(RR((1/a)*(-2*sqrt(B2)-
                    (MM[2][3])*(i3+j3*T)+Conjugate(d,(MM[2][3])*
                                    (i3+j3*T))))),RR((1/a)*(2*sqrt(B2)-
                                    (MM[2][3])*(i3+j3*T)+
```

```
          Conjugate(d,(MM[2][3])*(i3+j3*T))))  +1):
if RR(D[3][3]*(i3+j3*T)^2+D[2][2]*
(i2+j2*T+MM[2][3]*(i3+j3*T))^2
       + Conjugate(d,D[3][3]*(i3+j3*T)^2+
       D[2][2]*(i2+j2*T+MM[2][3]*(i3+j3*T))^2 )) <= n:
    for i1 in range(floor(RR(-sqrt(RR(B1))+
    (1/a)*((TB)*(MM[1][2]*(i2+j2*T)+
    MM[1][3]*(i3+j3*T))-
                  Conjugate(d,(TB)*(MM[1][2]*
                  (i2+j2*T)+MM[1][3]*(i3+j3*T)))))),
                  RR(sqrt(RR(B1))
                  +(1/a)*((TB)*(MM[1][2]*(i2+j2*T)+
                  MM[1][3]*(i3+j3*T))-
                  Conjugate(d,(TB)*(MM[1][2]*(i2+j2*T)
                  +MM[1][3]*(i3+j3*T))))  +1):
        for j1 in range(floor(RR((1/a)*(-2*sqrt(B1)
        -(MM[1][2]*(i2+j2*T)+MM[1][3]*(i3+j3*T))
                     +Conjugate(d,(MM[1][2]*(i2+j2*T)
                     +MM[1][3]*(i3+j3*T))))  )),
                     RR((1/a)*(2*sqrt(B1)-(MM[1][2]*(i2+j2*T)
                     +MM[1][3]*(i3+j3*T))+
```

```
          Conjugate(d,(MM[1][2]*(i2+j2*T)+
          MM[1][3]*(i3+j3*T))))  )+1):
if RR(D[1][1]*(i1+j1*T+MM[1][2]*(i2+j2*T)
+MM[1][3]*(i3+j3*T))^2+D[3][3]*(i3+j3*T)^2
    +D[2][2]*(i2+j2*T+MM[2][3]*(i3+j3*T))^2 +
    Conjugate(d,D[1][1]*(i1+j1*T+
    MM[1][2]*(i2+j2*T)+MM[1][3]*(i3+j3*T))^2+
    D[3][3]*(i3+j3*T)^2+D[2][2]*
    (i2+j2*T+MM[2][3]*(i3+j3*T))^2)) <= n:

    for i0 in range(floor(RR(-sqrt(B0)+
    (1/a)*(TB*(MM[0][1]*(i1+j1*T)+
    MM[0][2]*(i2+j2*T)
                +MM[0][3]*(i3+j3*T))-
                Conjugate(d,TB*(MM[0][1]*
                (i1+j1*T)+MM[0][2]*(i2+j2*T)
                +MM[0][3]*(i3+j3*T)))))   ),
                RR(sqrt(B0)+(1/a)*
                (TB*(MM[0][1]*(i1+j1*T)+MM[0][2]*(i2+j2*T)+
                MM[0][3]*(i3+j3*T))-
                Conjugate(d,TB*(MM[0][1]*
```

```
          (i1+j1*T)+MM[0][2]*(i2+j2*T)
          +MM[0][3]*(i3+j3*T)))))
           +1):
for j0 in range(floor(RR((1/a)*
(-2*sqrt(B0)-(MM[0][1]*(i1+j1*T)
+MM[0][2]*(i2+j2*T)
             +MM[0][3]*(i3+j3*T)) +
              Conjugate(d, (MM[0][1]*(i1+j1*T)
              +MM[0][2]*(i2+j2*T)
             +MM[0][3]*(i3+j3*T))))  ),
             RR((1/a)*(2*sqrt(B0)-
             (MM[0][1]*(i1+j1*T)+
             MM[0][2]*(i2+j2*T)+
             MM[0][3]*(i3+j3*T)) +
             Conjugate(d, (MM[0][1]*(i1+j1*T)
             +MM[0][2]*(i2+j2*T)+
             MM[0][3]*(i3+j3*T))))) +1):
    if List.count(
    integer_coefficients_of_representation(
    d,K(FQ.subs(z0 =
    i0 + ((1+a)/2)*j0, z1 = i1 + ((1+a)/2)*j1,
```

```
                                                    z2 = i2 + ((1+a)/2)*j2,
                                                   z3 = i3 + ((1+a)/2)*j3)))) >0:
                                                       Counting[List.index
                                                       (integer_coefficients_of_
                                                       representation(d,
                                                       K(FQ.subs(z0 = i0 + T*j0, z1 = i1 + T*j1,
                                                        z2 = i2 + T*j2, z3 = i3 + T*j3))))] =
                                                        Counting[List.index
                                                        (integer_coefficients_of_
                                                        representation(d,
                                                        K(FQ.subs(z0 = i0 + T*j0, z1 = i1 +
                                                        T*j1, z2 = i2 + T*j2, z3 = i3 + T*j3))))] + 1
                                                       counter +=1

    print Counting

else:
    for i3 in range(floor(-sqrt(RR(B3/2))),sqrt(RR(B3/2)) +1):
        for j3 in range(floor(-sqrt(RR(B3*DD))),sqrt(RR(B3*DD)) +1):
            if RR(D[3][3]*(i3+j3*a)^2+Conjugate(d,D[3][3]*(i3+j3*a)^2)) <= n:
                for i2 in range(floor(RR(-sqrt(RR(B2))-(1/2)*(MM[2][3]*(i3+j3*a)+
```

```
          Conjugate(d, MM[2][3]*(i3+j3*a))))),RR(sqrt(RR(B2))-(1/2)*
          (MM[2][3]*(i3+j3*a)
         +Conjugate(d, MM[2][3]*(i3+j3*a)))) +1):
for j2 in range(RR((1/(2*a))*(-2*sqrt(RR(B2))-
MM[2][3]*(i3+j3*a)+
             Conjugate(d, MM[2][3]*(i3+j3*a)))),RR((1/(2*a))*
             (2*sqrt(RR(B2))-MM[2][3]*(i3+j3*a)
             +Conjugate(d, MM[2][3]*(i3+j3*a)))) +1):
    if RR(D[2][2]*(i2+j2*a+MM[2][3]*(i3+j3*a))^2+
    D[3][3]*(i3+j3*a)^2 +
       Conjugate(d, D[2][2]*(i2+j2*a+MM[2][3]*(i3+j3*a))^2
       +D[3][3]*(i3+j3*a)^2) ) <=n:
        for i1 in range(floor(RR(-sqrt(RR(B1))-(1/2)*
        (MM[1][2]*(i2+j2*a)+MM[1][3]*(i3+j3*a+
                     Conjugate(d,MM[1][2]*(i2+j2*a)+
                     MM[1][3]*(i3+j3*a))))),RR(sqrt(RR(B1))-
                     (1/2)*(MM[1][2]*(i2+j2*a)+MM[1][3]*(i3+j3*a)+
                     Conjugate(d,MM[1][2]*(i2+j2*a)+
                     MM[1][3]*(i3+j3*a)))) +1):
            for j1 in range(RR((1/(2*a))*(-2*sqrt(RR(B1))-
            MM[1][2]*(i2+j2*a)-MM[1][3]*(i3+j3*a)
```

```
          +Conjugate(d, MM[1][2]*(i2+j2*a)+
          MM[1][3]*(i3+j3*a)))),
          RR((1/(2*a))*(2*sqrt(RR(B1))-MM[1][2]*(i2+j2*a)-
          MM[1][3]*(i3+j3*a)+
          Conjugate(d, MM[1][2]*(i2+j2*a)+
          MM[1][3]*(i3+j3*a)))) +1):
if RR(D[1][1]*(i1+j1*a+MM[1][2]*(i2+j2*a)+
MM[1][3]*(i3+j3*a))^2+
   D[2][2]*(i2+j2*a+MM[2][3]*(i3+j3*a))^2+
   D[3][3]*(i3+j3*a)^2 +
   Conjugate(d, D[1][1]*(i1+j1*a+
   MM[1][2]*(i2+j2*a)+MM[1][3]*(i3+j3*a))^2+
   D[2][2]*(i2+j2*a+MM[2][3]*(i3+j3*a))^2+
   D[3][3]*(i3+j3*a)^2) ) <=n:
    for i0 in range(floor(RR(-sqrt(RR(B0))-
    (1/2)*(MM[0][1]*(i1+j1*a)+
                  MM[0][2]*(i2+j2*a)+MM[0][3]*(i3+j3*a)+
                  Conjugate(d,MM[0][1]*(i1+j1*a)+
                  MM[0][2]*(i2+j2*a)+MM[0][3]*(i3+j3*a))))),
                  RR(sqrt(RR(B0))-(1/2)*(MM[0][1]*
                  (i1+j1*a)+MM[0][2]*(i2+j2*a)+
```

```
            MM[0][3]*(i3+j3*a)
        +Conjugate(d,MM[0][1]*(i1+j1*a)+
        MM[0][2]*(i2+j2*a)+MM[0][3]*
        (i3+j3*a)))) +1):
for j0 in range(RR((1/(2*a))*(-2*sqrt(RR(B0))-
MM[0][1]*(i1+j1*a)-
              MM[0][2]*(i2+j2*a)-MM[0][3]*(i3+j3*a)+
              Conjugate(d, MM[0][1]*(i1+j1*a)+
              MM[0][2]*(i2+j2*a)+MM[0][3]*(i3+j3*a)))),
              RR((1/(2*a))*(2*sqrt(RR(B0))-
              MM[0][1]*(i1+j1*a)-MM[0][2]*(i2+j2*a)-
              MM[0][3]*(i3+j3*a)+Conjugate(d,
              MM[0][1]*(i1+j1*a)+MM[0][2]*(i2+j2*a)+
              MM[0][3]*(i3+j3*a)))) +1):
    if List.count(integer_coefficients_of_representation
    (d,K(FQ.subs(
        z0 = i0 + a*j0, z1 = i1 + a*j1, z2 = i2 + a*j2,
         z3 = i3 + a*j3)))) >0:
        Counting[List.index(integer_coefficients_of_
        representation(d,K(FQ.subs(
        z0 = i0 + a*j0, z1 = i1 + a*j1, z2 = i2 + a*j2,
```

```
                                                    z3 = i3 + a*j3))))] =
                                                    Counting[List.index(integer_coefficients_
                                                    of_representation(d,K(FQ.subs(
                                                    z0 = i0 + a*j0, z1 = i1 + a*j1,
                                                    z2 = i2 + a*j2, z3 = i3 + a*j3))))] + 1
                                                    counter +=1

        print Counting


def Individual_Theta_Computation(d,Q,m):
    """
    For the particular m you feed, this will return the set of all oct-uples
    (i0,j0,i1,j1,i2,j2,i3,j3)
    which represent m under the form Q over QQ(sqrt(d))

    EXAMPLES:
    ======
    sage: x = var('x')
    sage: K.<a>=NumberField(x^2-2,embedding=1)
    sage: R=Matrix(K,4,[1,0,0,0,  0,1,0,0,  0,0,1,0,  0,0,0,8])
```

```
sage: Individual_Theta_Computation(2,R,a+2)
[]

sage: Individual_Theta_Computation(2,R,3)
[(-1, 0, -1, 0, -1, 0, 0, 0), (1, 0, -1, 0, -1, 0, 0, 0), (0, 0, 0, -1, -1, 0, 0, 0),
(0, -1, 0, 0, -1, 0, 0, 0), (0, 1, 0, 0, -1, 0, 0, 0), (0, 0, 0, 1, -1, 0, 0, 0),
(-1, 0, 1, 0, -1, 0, 0, 0), (1, 0, 1, 0, -1, 0, 0, 0), (0, 0, -1, 0, 0, -1, 0, 0),
(-1, 0, 0, 0, 0, -1, 0, 0), (1, 0, 0, 0, 0, -1, 0, 0), (0, 0, 1, 0, 0, -1, 0, 0),
(0, -1, -1, 0, 0, 0, 0, 0), (0, 1, -1, 0, 0, 0, 0, 0), (-1, 0, 0, -1, 0, 0, 0, 0),
(1, 0, 0, -1, 0, 0, 0, 0), (-1, 0, 0, 1, 0, 0, 0, 0), (1, 0, 0, 1, 0, 0, 0, 0),
(0, -1, 1, 0, 0, 0, 0, 0), (0, 1, 1, 0, 0, 0, 0, 0), (0, 0, -1, 0, 0, 1, 0, 0),
(-1, 0, 0, 0, 0, 1, 0, 0), (1, 0, 0, 0, 0, 1, 0, 0), (0, 0, 1, 0, 0, 1, 0, 0),
(-1, 0, -1, 0, 1, 0, 0, 0), (1, 0, -1, 0, 1, 0, 0, 0), (0, 0, 0, -1, 1, 0, 0, 0),
(0, -1, 0, 0, 1, 0, 0, 0), (0, 1, 0, 0, 1, 0, 0, 0), (0, 0, 0, 1, 1, 0, 0, 0),
(-1, 0, 1, 0, 1, 0, 0, 0), (1, 0, 1, 0, 1, 0, 0, 0)]

======
Below is code that returns the icosian group (c.f., Conway and Sloane) of 120 elements.
sage: x = var('x')
sage: K.<a>=NumberField(x^2-5,embedding=1)
sage: R=Matrix(K,4,[1,1,1,(1+a)/2,  0,1,(1+a)/2,(1+a)/2,  0,0,1,(1+a)/2,  0,0,0,1])
```

```
sage: UNITS=Individual_Theta_Computation(5,R,1)
sage: UNITS
[(1, 0, -1, 1, -1, 1, -1, -1), (2, 0, -1, 1, -1, 1, -1, -1), (1, 0, 0, 1, -1, 1, -1, -1),
(0, 1, 0, 1, 0, 0, -1, -1), (1, 0, 0, 1, 0, 0, -1, -1), (0, 1, 1, 0, 0, 0, -1, -1),
(1, 0, -1, 1, 0, 1, -1, -1), (0, 1, 0, 0, 0, 1, -1, -1), (1, 0, 0, 0, 0, 1, -1, -1),
(0, 1, 0, 0, 1, 0, -1, -1), (-1, 1, 1, 0, 1, 0, -1, -1), (0, 1, 1, 0, 1, 0, -1, -1),
(1, 0, -1, 1, -1, 0, -1, 0), (0, 0, 0, 1, -1, 0, -1, 0), (1, 0, 0, 1, -1, 0, -1, 0),
(1, 0, -1, 0, -1, 1, -1, 0), (1, -1, -1, 1, -1, 1, -1, 0), (1, 0, -1, 1, -1, 1, -1, 0),
 (0, 0, 0, 0, 0, 0, -1, 0), (0, 1, 0, 0, 0, 0, -1, 0), (0, 0, 0, 1, 0, 0, -1, 0),
 (0, 0, -1, 0, 0, 1, -1, 0), (1, 0, -1, 0, 0, 1, -1, 0), (0, 0, 0, 0, 0, 1, -1, 0),
 (0, 1, 1, 0, 1, 0, 0, -2), (1, 0, 0, 1, -1, 0, 0, -1), (0, 1, 0, 0, 0, 0, 0, -1),
 (1, 0, 0, 0, 0, 0, 0, -1), (0, 0, 0, 1, 0, 0, 0, -1), (1, 0, 0, 1, 0, 0, 0, -1),
(0, 0, 1, 0, 0, 0, 0, -1), (0, 1, 1, 0, 0, 0, 0, -1), (1, 0, -1, 0, 0, 1, 0, -1),
 (0, 0, 0, 0, 0, 1, 0, -1), (1, 0, 0, 0, 0, 1, 0, -1), (-1, 1, 1, 0, 1, -1, 0, -1),
  (0, 1, 1, 0, 1, -1, 0, -1), (-1, 1, 2, 0, 1, -1, 0, -1), (0, 0, 0, 0, 1, 0, 0, -1),
(0, 1, 0, 0, 1, 0, 0, -1), (-1, 1, 1, -1, 1, 0, 0, -1), (0, 1, 1, -1, 1, 0, 0, -1),
(-1, 1, 1, 0, 1, 0, 0, -1), (0, 0, 1, 0, 1, 0, 0, -1), (-1, 1, 1, -1, 2, 0, 0, -1),
(1, -1, -1, 1, -1, 0, 0, 0), (1, 0, -1, 1, -1, 0, 0, 0), (0, 0, 0, 0, -1, 0, 0, 0),
 (1, 0, 0, 0, -1, 0, 0, 0), (0, 0, 0, 1, -1, 0, 0, 0), (1, -1, 0, 1, -1, 0, 0, 0),
```

(1, -1, -1, 0, -1, 1, 0, 0), (1, 0, -1, 0, -1, 1, 0, 0), (1, -1, -1, 1, -1, 1, 0, 0),
(0, 0, 0, 1, 0, -1, 0, 0), (-1, 1, 1, 0, 0, -1, 0, 0), (0, 0, 1, 0, 0, -1, 0, 0),
(0, 0, -1, 0, 0, 0, 0, 0), (1, 0, -1, 0, 0, 0, 0, 0), (-1, 0, 0, 0, 0, 0, 0, 0),
(1, 0, 0, 0, 0, 0, 0, 0), (-1, 0, 1, 0, 0, 0, 0, 0), (0, 0, 1, 0, 0, 0, 0, 0),
(0, 0, -1, 0, 0, 1, 0, 0), (1, -1, -1, 0, 0, 1, 0, 0),(0, 0, 0, -1, 0, 1, 0, 0),
(-1, 1, 1, -1, 1, -1, 0, 0), (-1, 0, 1, 0, 1, -1, 0, 0), (-1, 1, 1, 0, 1, -1, 0, 0),
(-1, 1, 0, -1, 1, 0, 0, 0), (0, 0, 0, -1, 1, 0, 0, 0), (-1, 0, 0, 0, 1, 0, 0, 0),
(0, 0, 0, 0, 1, 0, 0, 0), (-1, 0, 1, -1, 1, 0, 0, 0), (-1, 1, 1, -1, 1, 0, 0, 0),
(1, -1, -1, 1, -2, 0, 0, 1), (0, 0, -1, 0, -1, 0, 0, 1), (1, -1, -1, 0, -1, 0, 0, 1),
(0, -1, -1, 1, -1, 0, 0, 1), (1, -1, -1, 1, -1, 0, 0, 1), (0, -1, 0, 0, -1, 0, 0, 1),
(0, 0, 0, 0, -1, 0, 0, 1), (1, -1, -2, 0, -1, 1, 0, 1), (0, -1, -1, 0, -1, 1, 0, 1),
(1, -1, -1, 0, -1, 1, 0, 1), (-1, 0, 0, 0, 0, -1, 0, 1), (0, 0, 0, 0, 0, -1, 0, 1),
(-1, 0, 1, 0, 0, -1, 0, 1), (0, -1, -1, 0, 0, 0, 0, 1), (0, 0, -1, 0, 0, 0, 0, 1),
(-1, 0, 0, -1, 0, 0, 0, 1), (0, 0, 0, -1, 0, 0, 0, 1), (-1, 0, 0, 0, 0, 0, 0, 1),
(0, -1, 0, 0, 0, 0, 0, 1), (-1, 0, 0, -1, 1, 0, 0, 1), (0, -1, -1, 0, -1, 0, 0, 2),
(0, 0, 0, 0, 0, -1, 1, 0), (-1, 0, 1, 0, 0, -1, 1, 0), (0, 0, 1, 0, 0, -1, 1, 0),
(0, 0, 0, -1, 0, 0, 1, 0), (0, -1, 0, 0, 0, 0, 1, 0), (0, 0, 0, 0, 0, 0, 1, 0),
(-1, 0, 1, -1, 1, -1, 1, 0), (-1, 1, 1, -1, 1, -1, 1, 0),
(-1, 0, 1, 0, 1, -1, 1, 0), (-1, 0, 0, -1, 1, 0, 1, 0), (0, 0, 0, -1, 1, 0, 1, 0),
(-1, 0, 1, -1, 1, 0, 1, 0), (0, -1, -1, 0, -1, 0, 1, 1), (1, -1, -1, 0, -1, 0, 1, 1),
(0, -1, 0, 0, -1, 0, 1, 1), (-1, 0, 0, 0, 0, -1, 1, 1), (0, -1, 0, 0, 0, -1, 1, 1),

```
 (-1, 0, 1, -1, 0, -1, 1, 1), (0, -1, -1, 0, 0, 0, 1, 1),
(-1, 0, 0, -1, 0, 0, 1, 1), (0, -1, 0, -1, 0, 0, 1, 1), (-1, 0, 0, -1, 1, -1, 1, 1),
(-2, 0, 1, -1, 1, -1, 1, 1), (-1, 0, 1, -1, 1, -1, 1, 1)]



"""
assert d.is_squarefree() == True
assert d > 0

assert is_totally_positive_definite_quaternary(d,Q) == True

x = QQ['x'].0
K.<a> = NumberField(x^2-d,embedding=1)
OK = K.ring_of_integers()



QasQF = QuadraticForm(K,(Q+Q.transpose())/2)
FQ = QasQF.polynomial(names='z')
Diag = (QasQF.rational_diagonal_form()).Gram_matrix()
D = 2*Diag
```

```
MM = change_of_basis_return(d,Q)
n = RR(K(m).trace())
List = List_for_all_Theta(d,Q,m)
L=[]

B3 = K(m/D[3][3]).trace()
B2 = K(m/D[2][2]).trace()
B1 = K(m/D[1][1]).trace()
B0 = K(m/D[0][0]).trace()
```



```
DD = find_D(d)
counter = 0
Counting = [0 for k in range(len(List))]
if d%4 ==1:

    T=(1+a)/2
    TB = Conjugate(d, T)
    IBT = 1/((1/T)-(1/TB))
    for i3 in range(floor(-sqrt(RR(B3))), sqrt(RR(B3)) +1):
```

```
for j3 in range(floor(-sqrt(RR(B3*DD))), sqrt(RR(B3*DD))+1):
    if RR(K(D[3][3]*(i3+j3*T)^2).trace()) <= n:
        for i2 in range(floor(RR(-sqrt(RR(B2))+(1/a)*
        ((TB)*MM[2][3]*(i3+j3*T)-
                    Conjugate(d,(TB)*MM[2][3]*(i3+j3*T)))),
                    sqrt(RR(B2))+(1/a)*((TB)*MM[2][3]*(i3+j3*T)-
                    Conjugate(d,(TB)*MM[2][3]*(i3+j3*T))) +1):
            for j2 in range(floor(RR((1/a)*(-2*sqrt(B2)-
            (MM[2][3])*(i3+j3*T)+
                        Conjugate(d,(MM[2][3])*(i3+j3*T)))),
                        RR((1/a)*(2*sqrt(B2)-(MM[2][3])*(i3+j3*T)+
                        Conjugate(d,(MM[2][3])*(i3+j3*T))) +1):
                if RR(D[3][3]*(i3+j3*T)^2+D[2][2]*(i2+j2*T+
                MM[2][3]*(i3+j3*T))^2 +
                    Conjugate(d,D[3][3]*(i3+j3*T)^2+
                    D[2][2]*(i2+j2*T+MM[2][3]*(i3+j3*T))^2 )) <= n:
                    for i1 in range(floor(RR(-sqrt(RR(B1))+(1/a)*
                    ((TB)*(MM[1][2]*(i2+j2*T)+MM[1][3]*(i3+j3*T))-
                                Conjugate(d,(TB)*(MM[1][2]*(i2+j2*T)
                                +MM[1][3]*(i3+j3*T))))),
                                RR(sqrt(RR(B1))+(1/a)*((TB)*
```

```
            (MM[1][2]*(i2+j2*T)+MM[1][3]*(i3+j3*T))-
            Conjugate(d,(TB)*(MM[1][2]*(i2+j2*T)
            +MM[1][3]*(i3+j3*T))))) +1):
for j1 in range(floor(RR((1/a)*(-2*sqrt(B1)
-(MM[1][2]*(i2+j2*T)+MM[1][3]*(i3+j3*T))
              +Conjugate(d,(MM[1][2]*(i2+j2*T)
              +MM[1][3]*(i3+j3*T))))  )),
              RR((1/a)*(2*sqrt(B1)-(MM[1][2]*
              (i2+j2*T)+MM[1][3]*(i3+j3*T))+
              Conjugate(d,(MM[1][2]*(i2+j2*T)+
              MM[1][3]*(i3+j3*T))))  )+1):
    if RR(D[1][1]*(i1+j1*T+MM[1][2]*(i2+j2*T)
    +MM[1][3]*(i3+j3*T))^2+D[3][3]*(i3+j3*T)^2+
       D[2][2]*(i2+j2*T+MM[2][3]*(i3+j3*T))^2 +
       Conjugate(d,D[1][1]*(i1+j1*T+
       MM[1][2]*(i2+j2*T)+MM[1][3]*(i3+j3*T))^2+
       D[3][3]*(i3+j3*T)^2+D[2][2]*(i2+j2*T+
       MM[2][3]*(i3+j3*T))^2)) <= n:
        for i0 in range(floor(RR(-sqrt(B0)+(1/a)*
        (TB*(MM[0][1]*(i1+j1*T)+MM[0][2]*(i2+j2*T)
                    +MM[0][3]*(i3+j3*T))-
```

```
            Conjugate(d,TB*(MM[0][1]*(i1+j1*T)
            +MM[0][2]*(i2+j2*T)
            +MM[0][3]*(i3+j3*T)))))   ),
            RR(sqrt(B0)+(1/a)*(TB*(MM[0][1]*
            (i1+j1*T)+MM[0][2]*(i2+j2*T)+
            MM[0][3]*(i3+j3*T))-
            Conjugate(d,TB*(MM[0][1]*
            (i1+j1*T)+MM[0][2]*(i2+j2*T)+
            MM[0][3]*(i3+j3*T)))) +1):
for j0 in range(floor(RR((1/a)*(-2*
sqrt(B0)-(MM[0][1]*(i1+j1*T)+
MM[0][2]*(i2+j2*T)
                +MM[0][3]*(i3+j3*T)) +
                 Conjugate(d, (MM[0][1]*
                 (i1+j1*T)+MM[0][2]*(i2+j2*T)
                +MM[0][3]*(i3+j3*T)))))  ),
                RR((1/a)*(2*sqrt(B0)-
                (MM[0][1]*(i1+j1*T)+
                MM[0][2]*(i2+j2*T)+
                MM[0][3]*(i3+j3*T)) +
                Conjugate(d, (MM[0][1]*
```

```
                                                    (i1+j1*T)+MM[0][2]*(i2+j2*T)+
                                                    MM[0][3]*(i3+j3*T))))) +1):
                                                if integer_coefficients_of_
                                                representation(d,K(FQ.subs(
                                                z0 = i0 + ((1+a)/2)*j0,
                                                z1 = i1 + ((1+a)/2)*j1,
                                                z2 = i2 + ((1+a)/2)*j2,
                                                z3 = i3 + ((1+a)/2)*j3))) ==
                                                 integer_coefficients_of_
                                                 representation(d,K(m)):
                                                    L.append((i0,j0,i1,j1,i2,j2,i3,j3))

    return L

else:
    for i3 in range(floor(-sqrt(RR(B3/2))),sqrt(RR(B3/2)) +1):
        for j3 in range(floor(-sqrt(RR(B3*DD))),sqrt(RR(B3*DD)) +1):
            if RR(D[3][3]*(i3+j3*a)^2+Conjugate(d,D[3][3]*(i3+j3*a)^2)) <= n:
                for i2 in range(floor(RR(-sqrt(RR(B2))-(1/2)*(MM[2][3]*(i3+j3*a)+
                                    Conjugate(d, MM[2][3]*(i3+j3*a))))),
                                    RR(sqrt(RR(B2))-(1/2)*(MM[2][3]*(i3+j3*a)+
```

```
          Conjugate(d, MM[2][3]*(i3+j3*a)))) +1):
for j2 in range(RR((1/(2*a))*(-2*sqrt(RR(B2))-
MM[2][3]*(i3+j3*a)+
              Conjugate(d, MM[2][3]*(i3+j3*a)))),
              RR((1/(2*a))*(2*sqrt(RR(B2))-MM[2][3]*(i3+j3*a)
              +Conjugate(d, MM[2][3]*(i3+j3*a)))) +1):
    if RR(D[2][2]*(i2+j2*a+MM[2][3]*(i3+j3*a))^2+
    D[3][3]*(i3+j3*a)^2 +
        Conjugate(d, D[2][2]*(i2+j2*a+
        MM[2][3]*(i3+j3*a))^2+D[3][3]*(i3+j3*a)^2) ) <=n:

        for i1 in range(floor(RR(-sqrt(RR(B1))-
        (1/2)*(MM[1][2]*(i2+j2*a)+MM[1][3]*(i3+j3*a)
                    +Conjugate(d,MM[1][2]*(i2+j2*a)+
                    MM[1][3]*(i3+j3*a)))),
                    RR(sqrt(RR(B1))-(1/2)*(MM[1][2]*(i2+j2*a)
                    +MM[1][3]*(i3+j3*a)+
                    Conjugate(d,MM[1][2]*(i2+j2*a)+
                    MM[1][3]*(i3+j3*a)))) +1):

            for j1 in range(RR((1/(2*a))*(-2*sqrt(RR(B1))-
```

```
MM[1][2]*(i2+j2*a)-MM[1][3]*(i3+j3*a)
             +Conjugate(d, MM[1][2]*(i2+j2*a)
             +MM[1][3]*(i3+j3*a)))),
             RR((1/(2*a))*(2*sqrt(RR(B1))-
             MM[1][2]*(i2+j2*a)-MM[1][3]*(i3+j3*a)+
             Conjugate(d, MM[1][2]*(i2+j2*a)
             +MM[1][3]*(i3+j3*a)))) +1):
    if RR(D[1][1]*(i1+j1*a+MM[1][2]*(i2+j2*a)
    +MM[1][3]*(i3+j3*a))^2+
        D[2][2]*(i2+j2*a+MM[2][3]*(i3+j3*a))^2
        +D[3][3]*(i3+j3*a)^2 +
        Conjugate(d, D[1][1]*(i1+j1*a+
        MM[1][2]*(i2+j2*a)+MM[1][3]*(i3+j3*a))^2+
        D[2][2]*(i2+j2*a+MM[2][3]*(i3+j3*a))^2+
        D[3][3]*(i3+j3*a)^2) ) <=n:

        for i0 in range(floor(RR(-sqrt(RR(B0))-
        (1/2)*(MM[0][1]*(i1+j1*a)+MM[0][2]*(i2+j2*a)
                     +MM[0][3]*(i3+j3*a)+
                     Conjugate(d,MM[0][1]*(i1+j1*a)+
                     MM[0][2]*(i2+j2*a)+
```

```
            MM[0][3]*(i3+j3*a))))),RR(sqrt(RR(B0))-
            (1/2)*(MM[0][1]*(i1+j1*a)+
            MM[0][2]*(i2+j2*a)+MM[0][3]*(i3+j3*a)+
            Conjugate(d,MM[0][1]*(i1+j1*a)+
            MM[0][2]*(i2+j2*a)+MM[0][3]*(i3+j3*a)))) +1):

for j0 in range(RR((1/(2*a))*(-2*sqrt(RR(B0))
-MM[0][1]*(i1+j1*a)-MM[0][2]*(i2+j2*a)-
                MM[0][3]*(i3+j3*a)+Conjugate(d,
                MM[0][1]*(i1+j1*a)+MM[0][2]*(i2+j2*a)+
                MM[0][3]*(i3+j3*a))),RR((1/(2*a))*
                (2*sqrt(RR(B0))-MM[0][1]*(i1+j1*a)-
                MM[0][2]*(i2+j2*a)-MM[0][3]*(i3+j3*a)
                +Conjugate(d, MM[0][1]*(i1+j1*a)+
                MM[0][2]*(i2+j2*a)+MM[0][3]*(i3+j3*a)))) +1):
    if integer_coefficients_of_representation
    (d,K(FQ.subs(z0 = i0 + a*j0,
    z1 = i1 + a*j1, z2 = i2 + a*j2, z3 = i3 + a*j3)))
    ==integer_coefficients_of_representation(d, K(m)):
        L.append((i0,j0,i1,j1,i2,j2,i3,j3))
```

```
return L
```

A.2 Code for Hilbert Modular Forms

Below is code which computes Brandt matrices for spaces of Hilbert modular forms over $\mathbb{Q}(\sqrt{5})$ and trivial character, via Dembélé's method.

```
########################################
## This code will code the algorithms ##
## outlined in Dembele's paper on     ##
## quatenrionic modular forms and the ##
##Jacquet-Langlands correspondence    ##
##applied to the specific case of the ##
##Hamiltonian quaternion algebra      ##
##defined over K= QQ(sqrt(5)).        ##
########################################


####################################
## This section of code is        ##
## meant to allow basic           ##
## quaternion element maniuplation ##
####################################
def Representations_in_Maximal_Order_Octuple(m):
    """
    This considers a totally-positive integer m in O_K, for K=QQ(sqrt(5))
    and returns the elements of norm m in the unique maximal order
    of the Hamiltonian quaternion algebra over K.
```

```
Here, as in Dembele's paper, we take the basis of the order to be
e_1 = 1/2*(1-(1-a)/2*i+(1+a)/2*j)
e_2= 1/2*(-(1-a)/2*i+j+(1+a)/2*k)
e_3 = 1/2*((1+a)/2*i-(1-a)/2*j+k)
e_4 = 1/2*(i+(1+a)/2*j-(1-a)/2*k).

The output will be all (a_1,b_1,a_2,b_2,a_3,b_3,a_4,b_4) such that
the norm of (a_1+b_1*(1+a)/2)*e_1+...+(a_4+b_4*(1+a)/2)*e_4 is m.
=============
EXAMPLES:
=============
sage: Representations_in_Maximal_Order_Octuple(a)
Traceback (most recent call last):
...
AssertionError
=============
=============

sage: Representations_in_Maximal_Order_Octuple(1)
[(1, 0, -1, 1, -1, 1, -1, -1), (2, 0, -1, 1, -1, 1, -1, -1), (1, 0, 0, 1, -1, 1, -1, -1),
```

(0, 1, 0, 1, 0, 0, -1, -1), (1, 0, 0, 1, 0, 0, -1, -1), (0, 1, 1, 0, 0, 0, -1, -1),
(1, 0, -1, 1, 0, 1, -1, -1), (0, 1, 0, 0, 0, 1, -1, -1), (1, 0, 0, 0, 0, 1, -1, -1),
(0, 1, 0, 0, 1, 0, -1, -1), (-1, 1, 1, 0, 1, 0, -1, -1), (0, 1, 1, 0, 1, 0, -1, -1),
(1, 0, -1, 1, -1, 0, -1, 0), (0, 0, 0, 1, -1, 0, -1, 0), (1, 0, 0, 1, -1, 0, -1, 0),
(1, 0, -1, 0, -1, 1, -1, 0), (1, -1, -1, 1, -1, 1, -1, 0), (1, 0, -1, 1, -1, 1, -1, 0),
(0, 0, 0, 0, 0, 0, -1, 0), (0, 1, 0, 0, 0, 0, -1, 0), (0, 0, 0, 1, 0, 0, -1, 0),
(0, 0, -1, 0, 0, 1, -1, 0), (1, 0, -1, 0, 0, 1, -1, 0), (0, 0, 0, 0, 0, 1, -1, 0),
(0, 1, 1, 0, 1, 0, 0, -2), (1, 0, 0, 1, -1, 0, 0, -1), (0, 1, 0, 0, 0, 0, 0, -1),
(1, 0, 0, 0, 0, 0, 0, -1), (0, 0, 0, 1, 0, 0, 0, -1), (1, 0, 0, 1, 0, 0, 0, -1),
(0, 0, 1, 0, 0, 0, 0, -1), (0, 1, 1, 0, 0, 0, 0, -1), (1, 0, -1, 0, 0, 1, 0, -1),
(0, 0, 0, 0, 0, 1, 0, -1), (1, 0, 0, 0, 0, 1, 0, -1), (-1, 1, 1, 0, 1, -1, 0, -1),
(0, 1, 1, 0, 1, -1, 0, -1), (-1, 1, 2, 0, 1, -1, 0, -1), (0, 0, 0, 0, 1, 0, 0, -1),
(0, 1, 0, 0, 1, 0, 0, -1), (-1, 1, 1, -1, 1, 0, 0, -1), (0, 1, 1, -1, 1, 0, 0, -1),
(-1, 1, 1, 0, 1, 0, 0, -1), (0, 0, 1, 0, 1, 0, 0, -1), (-1, 1, 1, -1, 2, 0, 0, -1),
(1, -1, -1, 1, -1, 0, 0, 0), (1, 0, -1, 1, -1, 0, 0, 0), (0, 0, 0, 0, -1, 0, 0, 0),
(1, 0, 0, 0, -1, 0, 0, 0), (0, 0, 0, 1, -1, 0, 0, 0), (1, -1, 0, 1, -1, 0, 0, 0),
(1, -1, -1, 0, -1, 1, 0, 0), (1, 0, -1, 0, -1, 1, 0, 0), (1, -1, -1, 1, -1, 1, 0, 0),
(0, 0, 0, 1, 0, -1, 0, 0), (-1, 1, 1, 0, 0, -1, 0, 0), (0, 0, 1, 0, 0, -1, 0, 0),
(0, 0, -1, 0, 0, 0, 0, 0), (1, 0, -1, 0, 0, 0, 0, 0), (-1, 0, 0, 0, 0, 0, 0, 0),
(1, 0, 0, 0, 0, 0, 0, 0), (-1, 0, 1, 0, 0, 0, 0, 0), (0, 0, 1, 0, 0, 0, 0, 0),
(0, 0, -1, 0, 0, 1, 0, 0), (1, -1, -1, 0, 0, 1, 0, 0), (0, 0, 0, -1, 0, 1, 0, 0),

(-1, 1, 1, -1, 1, -1, 0, 0), (-1, 0, 1, 0, 1, -1, 0, 0), (-1, 1, 1, 0, 1, -1, 0, 0),
(-1, 1, 0, -1, 1, 0, 0, 0), (0, 0, 0, -1, 1, 0, 0, 0), (-1, 0, 0, 0, 1, 0, 0, 0),
(0, 0, 0, 0, 1, 0, 0, 0), (-1, 0, 1, -1, 1, 0, 0, 0), (-1, 1, 1, -1, 1, 0, 0, 0),
(1, -1, -1, 1, -2, 0, 0, 1), (0, 0, -1, 0, -1, 0, 0, 1), (1, -1, -1, 0, -1, 0, 0, 1),
(0, -1, -1, 1, -1, 0, 0, 1), (1, -1, -1, 1, -1, 0, 0, 1), (0, -1, 0, 0, -1, 0, 0, 1),
(0, 0, 0, 0, -1, 0, 0, 1), (1, -1, -2, 0, -1, 1, 0, 1), (0, -1, -1, 0, -1, 1, 0, 1),
(1, -1, -1, 0, -1, 1, 0, 1), (-1, 0, 0, 0, 0, -1, 0, 1), (0, 0, 0, 0, 0, -1, 0, 1),
(-1, 0, 1, 0, 0, -1, 0, 1), (0, -1, -1, 0, 0, 0, 0, 1), (0, 0, -1, 0, 0, 0, 0, 1),
(-1, 0, 0, -1, 0, 0, 0, 1), (0, 0, 0, -1, 0, 0, 0, 1), (-1, 0, 0, 0, 0, 0, 0, 1),
(0, -1, 0, 0, 0, 0, 0, 1), (-1, 0, 0, -1, 1, 0, 0, 1), (0, -1, -1, 0, -1, 0, 0, 2),
(0, 0, 0, 0, 0, -1, 1, 0), (-1, 0, 1, 0, 0, -1, 1, 0), (0, 0, 1, 0, 0, -1, 1, 0),
(0, 0, 0, -1, 0, 0, 1, 0), (0, -1, 0, 0, 0, 0, 1, 0), (0, 0, 0, 0, 0, 0, 1, 0),
(-1, 0, 1, -1, 1, -1, 1, 0), (-1, 1, 1, -1, 1, -1, 1, 0), (-1, 0, 1, 0, 1, -1, 1, 0),
(-1, 0, 0, -1, 1, 0, 1, 0), (0, 0, 0, -1, 1, 0, 1, 0), (-1, 0, 1, -1, 1, 0, 1, 0),
(0, -1, -1, 0, -1, 0, 1, 1), (1, -1, -1, 0, -1, 0, 1, 1), (0, -1, 0, 0, -1, 0, 1, 1),
(-1, 0, 0, 0, 0, -1, 1, 1), (0, -1, 0, 0, 0, -1, 1, 1), (-1, 0, 1, -1, 0, -1, 1, 1),
(0, -1, -1, 0, 0, 0, 1, 1), (-1, 0, 0, -1, 0, 0, 1, 1), (0, -1, 0, -1, 0, 0, 1, 1),
(-1, 0, 0, -1, 1, -1, 1, 1), (-2, 0, 1, -1, 1, -1, 1, 1), (-1, 0, 1, -1, 1, -1, 1, 1)]

==========

==========

```
    """
    x = QQ['x'].0
    K.<a> = NumberField(x^2-5,embedding=1)
    OK = K.ring_of_integers()

    assert K(m).is_totally_positive() == True

    R = Matrix(K,4,[1,1,1,(1+a)/2,  0,1,(1+a)/2, (1+a)/2, 0,0,1, (1+a)/2, 0,0,0,1])

    return Individual_Theta_Computation(5,R,m)
```



```
def Representations_in_Maximal_Order_Quadruple(m):
    """
    This considers a totally-positive integer m in O_K, for K=QQ(sqrt(5))
    and returns the elements of norm m in the unique maximal order
    of the Hamiltonian quaternion algebra over K.

    Here, as in Dembele's paper, we take the basis of the order to be
```

```
e_1 = 1/2*(1-(1-a)/2*i+(1+a)/2*j)

e_2= 1/2*(-(1-a)/2*i+j+(1+a)/2*k)

e_3 = 1/2*((1+a)/2*i-(1-a)/2*j+k)

e_4 = 1/2*(i+(1+a)/2*j-(1-a)/2*k).


The output will be all O_K integer tuples (k_1,k_2,k_3,k_4) such that

the norm of k_1*e_1+...+k_4*e_4 is m.


==============

EXAMPLES:

==============

sage: Representations_in_Maximal_Order_Quadruple(a)

Traceback (most recent call last):

...

AssertionError

==============

==============


sage: Representations_in_Maximal_Order_Quadruple(1)

[(1, 1/2*a - 1/2, 1/2*a - 1/2, -1/2*a - 3/2), (2, 1/2*a - 1/2, 1/2*a - 1/2, -1/2*a - 3/2),

(1, 1/2*a + 1/2, 1/2*a - 1/2, -1/2*a - 3/2), (1/2*a + 1/2, 1/2*a + 1/2, 0, -1/2*a - 3/2),
```

(1, 1/2*a + 1/2, 0, -1/2*a - 3/2), (1/2*a + 1/2, 1, 0, -1/2*a - 3/2),

(1, 1/2*a - 1/2, 1/2*a + 1/2, -1/2*a - 3/2), (1/2*a + 1/2, 0, 1/2*a + 1/2, -1/2*a - 3/2),

(1, 0, 1/2*a + 1/2, -1/2*a - 3/2), (1/2*a + 1/2, 0, 1, -1/2*a - 3/2),

(1/2*a - 1/2, 1, 1, -1/2*a - 3/2), (1/2*a + 1/2, 1, 1, -1/2*a - 3/2), (1, 1/2*a - 1/2, -1, -1),

(0, 1/2*a + 1/2, -1, -1), (1, 1/2*a + 1/2, -1, -1), (1, -1, 1/2*a - 1/2, -1),

(-1/2*a + 1/2, 1/2*a - 1/2, 1/2*a - 1/2, -1), (1, 1/2*a - 1/2, 1/2*a - 1/2, -1), (0, 0, 0, -1),

(1/2*a + 1/2, 0, 0, -1), (0, 1/2*a + 1/2, 0, -1), (0, -1, 1/2*a + 1/2, -1),

(1, -1, 1/2*a + 1/2, -1), (0, 0, 1/2*a + 1/2, -1), (1/2*a + 1/2, 1, 1, -a - 1),

(1, 1/2*a + 1/2, -1, -1/2*a - 1/2), (1/2*a + 1/2, 0, 0, -1/2*a - 1/2), (1, 0, 0, -1/2*a - 1/2),

(0, 1/2*a + 1/2, 0, -1/2*a - 1/2), (1, 1/2*a + 1/2, 0, -1/2*a - 1/2), (0, 1, 0, -1/2*a - 1/2),

(1/2*a + 1/2, 1, 0, -1/2*a - 1/2), (1, -1, 1/2*a + 1/2, -1/2*a - 1/2),

(0, 0, 1/2*a + 1/2, -1/2*a - 1/2), (1, 0, 1/2*a + 1/2, -1/2*a - 1/2),

(1/2*a - 1/2, 1, -1/2*a + 1/2, -1/2*a - 1/2), (1/2*a + 1/2, 1, -1/2*a + 1/2, -1/2*a - 1/2),

(1/2*a - 1/2, 2, -1/2*a + 1/2, -1/2*a - 1/2), (0, 0, 1, -1/2*a - 1/2),

(1/2*a + 1/2, 0, 1, -1/2*a - 1/2), (1/2*a - 1/2, -1/2*a + 1/2, 1, -1/2*a - 1/2),

(1/2*a + 1/2, -1/2*a + 1/2, 1, -1/2*a - 1/2), (1/2*a - 1/2, 1, 1, -1/2*a - 1/2),

(0, 1, 1, -1/2*a - 1/2), (1/2*a - 1/2, -1/2*a + 1/2, 2, -1/2*a - 1/2),

(-1/2*a + 1/2, 1/2*a - 1/2, -1, 0), (1, 1/2*a - 1/2, -1, 0), (0, 0, -1, 0), (1, 0, -1, 0),

(0, 1/2*a + 1/2, -1, 0), (-1/2*a + 1/2, 1/2*a + 1/2, -1, 0), (-1/2*a + 1/2, -1, 1/2*a - 1/2, 0),

(1, -1, 1/2*a - 1/2, 0), (-1/2*a + 1/2, 1/2*a - 1/2, 1/2*a - 1/2, 0),

(0, 1/2*a + 1/2, -1/2*a - 1/2, 0), (1/2*a - 1/2, 1, -1/2*a - 1/2, 0), (0, 1, -1/2*a - 1/2, 0),

(0, -1, 0, 0), (1, -1, 0, 0), (-1, 0, 0, 0), (1, 0, 0, 0), (-1, 1, 0, 0), (0, 1, 0, 0),

(0, -1, 1/2*a + 1/2, 0), (-1/2*a + 1/2, -1, 1/2*a + 1/2, 0), (0, -1/2*a - 1/2, 1/2*a + 1/2, 0),

(1/2*a - 1/2, -1/2*a + 1/2, -1/2*a + 1/2, 0), (-1, 1, -1/2*a + 1/2, 0),

(1/2*a - 1/2, 1, -1/2*a + 1/2, 0), (1/2*a - 1/2, -1/2*a - 1/2, 1, 0), (0, -1/2*a - 1/2, 1, 0),

(-1, 0, 1, 0), (0, 0, 1, 0), (-1, -1/2*a + 1/2, 1, 0), (1/2*a - 1/2, -1/2*a + 1/2, 1, 0),

(-1/2*a + 1/2, 1/2*a - 1/2, -2, 1/2*a + 1/2), (0, -1, -1, 1/2*a + 1/2),

(-1/2*a + 1/2, -1, -1, 1/2*a + 1/2), (-1/2*a - 1/2, 1/2*a - 1/2, -1, 1/2*a + 1/2),

(-1/2*a + 1/2, 1/2*a - 1/2, -1, 1/2*a + 1/2), (-1/2*a - 1/2, 0, -1, 1/2*a + 1/2),

(0, 0, -1, 1/2*a + 1/2), (-1/2*a + 1/2, -2, 1/2*a - 1/2, 1/2*a + 1/2),

(-1/2*a - 1/2, -1, 1/2*a - 1/2, 1/2*a + 1/2), (-1/2*a + 1/2, -1, 1/2*a - 1/2, 1/2*a + 1/2),

(-1, 0, -1/2*a - 1/2, 1/2*a + 1/2), (0, 0, -1/2*a - 1/2, 1/2*a + 1/2),

(-1, 1, -1/2*a - 1/2, 1/2*a + 1/2), (-1/2*a - 1/2, -1, 0, 1/2*a + 1/2), (0, -1, 0, 1/2*a + 1/2),

(-1, -1/2*a - 1/2, 0, 1/2*a + 1/2), (0, -1/2*a - 1/2, 0, 1/2*a + 1/2), (-1, 0, 0, 1/2*a + 1/2),

(-1/2*a - 1/2, 0, 0, 1/2*a + 1/2), (-1, -1/2*a - 1/2, 1, 1/2*a + 1/2),

(-1/2*a - 1/2, -1, -1, a + 1), (0, 0, -1/2*a - 1/2, 1), (-1, 1, -1/2*a - 1/2, 1),

(0, 1, -1/2*a - 1/2, 1), (0, -1/2*a - 1/2, 0, 1), (-1/2*a - 1/2, 0, 0, 1), (0, 0, 0, 1),

(-1, -1/2*a + 1/2, -1/2*a + 1/2, 1), (1/2*a - 1/2, -1/2*a + 1/2, -1/2*a + 1/2, 1),

(-1, 1, -1/2*a + 1/2, 1), (-1, -1/2*a - 1/2, 1, 1), (0, -1/2*a - 1/2, 1, 1),

(-1, -1/2*a + 1/2, 1, 1), (-1/2*a - 1/2, -1, -1, 1/2*a + 3/2), (-1/2*a + 1/2, -1, -1, 1/2*a + 3/2),

(-1/2*a - 1/2, 0, -1, 1/2*a + 3/2), (-1, 0, -1/2*a - 1/2, 1/2*a + 3/2),

(-1/2*a - 1/2, 0, -1/2*a - 1/2, 1/2*a + 3/2), (-1, -1/2*a + 1/2, -1/2*a - 1/2, 1/2*a + 3/2),

```
(-1/2*a - 1/2, -1, 0, 1/2*a + 3/2), (-1, -1/2*a - 1/2, 0, 1/2*a + 3/2),
(-1/2*a - 1/2, -1/2*a - 1/2, 0, 1/2*a + 3/2), (-1, -1/2*a - 1/2, -1/2*a + 1/2, 1/2*a + 3/2),
(-2, -1/2*a + 1/2, -1/2*a + 1/2, 1/2*a + 3/2), (-1, -1/2*a + 1/2, -1/2*a + 1/2, 1/2*a + 3/2)]

"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

T = (a+1)/2

assert K(m).is_totally_positive() == True

R = Matrix(K,4,[1,1,1,T,  0,1,T,T, 0,0,1,T, 0,0,0,1])

I = Individual_Theta_Computation(5,R,m)
Set = []
for i in I:
    Set.append((i[0]+i[1]*T, i[2]+i[3]*T, i[4]+i[5]*T, i[6]+i[7]*T))
return Set
```

```
def Quaternion_Basis_Representation((a0,a1,a2,a3,a4,a5,a6,a7)):
    """
    This takes an octuple that is presumed to be written in the basis for
    the e_i's of the order, and rewrites it as a tuple in the basis
    <1,i,j,k> for the quaternion algebra.

    Again, the e_i basis is taken to be
    e_1 = 1/2*(1-(1-a)/2*i+(1+a)/2*j)
    e_2= 1/2*(-(1-a)/2*i+j+(1+a)/2*k)
    e_3 = 1/2*((1+a)/2*i-(1-a)/2*j+k)
    e_4 = 1/2*(i+(1+a)/2*j-(1-a)/2*k).

    The output will be (b0,b1,b2,b3,b4,b5,b6,b7) where
    (a0+a1*(1+a)/2)*e_1+...+(a6+a7*(1+a)/2)e_4 = (b0+b1*(1+a)/2) +...+(b6+b7*(1+a)/2)*k
    =============
    EXAMPLES:
    =============
    sage: Quaternion_Basis_Representation((1,13,1/3,0,1,2,-1,9))
```

```
(1/2, 13/2, 19/3, 20/3, 35/3, 23/2, 11/2, 2/3)
==============
==============
sage: Quaternion_Basis_Representation((1,0,0,0,0,0,0,0))
(1/2, 0, -1/2, 1/2, 0, 1/2, 0, 0)
==============
==============
"""
assert type(QQ(a0)) == type(QQ(1))
assert type(QQ(a1)) == type(QQ(1))
assert type(QQ(a2)) == type(QQ(1))
assert type(QQ(a3)) == type(QQ(1))
assert type(QQ(a4)) == type(QQ(1))
assert type(QQ(a5)) == type(QQ(1))
assert type(QQ(a6)) == type(QQ(1))
assert type(QQ(a7)) == type(QQ(1))

A0 = (1/2)*a0
A1 = (1/2)*a1
A2 = (1/2)*(a1-a0+a3-a2+a5+a6)
A3 = (1/2)*(a0+a2+a4+a7+a5)
```

```
        A4 = (1/2)*(a1+a2+a5-a4+a7)

        A5 = (1/2)*(a0+a1+a3+a4+a6+a7)

        A6 = (1/2)*(a3+a4+a7-a6)

        A7 = (1/2)*(a2+a3+a5+a6)


        A = (A0,A1,A2,A3,A4,A5,A6,A7)


        return A




def Representations_in_Max_Order_Trad_Basis(m):
    """
    This takes all elements of Representations_in_Maximal_Order_Octuple(m)
    and rewrites them in terms of Quaternion_Basis_Respresentation.

    =============
    EXAMPLES:
    =============

    sage: Representations_in_Max_Order_Trad_Basis(1)
```

[(1/2, 0, 1/2, -1/2, 0, -1/2, 0, 0), (1, 0, 0, 0, 0, 0, 0, 0),
(1/2, 0, 0, 0, 1/2, -1/2, 0, 1/2),
(0, 1/2, 1/2, -1/2, 0, 0, 1/2, 0), (1/2, 0, -1/2, 0, -1/2, 0, 1/2, 0),
(0, 1/2, -1/2, 0, 1/2, -1/2, 0, 0), (1/2, 0, 1/2, 0, -1/2, 0, 1/2, 0),
(0, 1/2, 1/2, 0, 1/2, -1/2, 0, 0), (1/2, 0, -1/2, 1/2, 0, -1/2, 0, 0),
(0, 1/2, 0, 0, -1/2, 0, 1/2, -1/2), (-1/2, 1/2, 0, 0, 0, -1/2, 1/2, 0),
(0, 1/2, -1/2, 1/2, 0, 0, 1/2, 0), (1/2, 0, 0, -1/2, 0, 0, 1/2, -1/2),
(0, 0, 0, -1/2, 1/2, -1/2, 1/2, 0), (1/2, 0, -1/2, 0, 1/2, 0, 1/2, 0),
(1/2, 0, 0, 0, 1/2, -1/2, 0, -1/2), (1/2, -1/2, 0, 0, 0, -1/2, 1/2, 0),
(1/2, 0, 1/2, 0, 1/2, 0, 1/2, 0), (0, 0, -1/2, 0, 0, -1/2, 1/2, -1/2),
(0, 1/2, 0, 0, 1/2, 0, 1/2, -1/2), (0, 0, 0, 0, 0, 0, 1, 0),
(0, 0, 1/2, 0, 0, -1/2, 1/2, -1/2), (1/2, 0, 0, 1/2, 0, 0, 1/2, -1/2),
(0, 0, 0, 1/2, 1/2, -1/2, 1/2, 0), (0, 1/2, 0, 0, -1/2, 0, -1/2, 1/2),
(1/2, 0, 0, -1/2, 0, 0, -1/2, 1/2), (0, 1/2, 1/2, -1/2, 0, 0, -1/2, 0),
(1/2, 0, -1/2, 0, -1/2, 0, -1/2, 0), (0, 0, 1/2, -1/2, -1/2, 0, 0, 1/2),
(1/2, 0, 0, 0, -1/2, 1/2, 0, 1/2), (0, 0, -1/2, 0, 0, -1/2, -1/2, 1/2),
(0, 1/2, 0, 0, 1/2, 0, -1/2, 1/2), (1/2, 0, 1/2, 0, -1/2, 0, -1/2, 0),
(0, 0, 1/2, 0, 0, -1/2, -1/2, 1/2), (1/2, 0, 0, 1/2, 0, 0, -1/2, 1/2),
(-1/2, 1/2, 0, -1/2, -1/2, 0, 0, 0), (0, 1/2, -1/2, 0, -1/2, 1/2, 0, 0),
(-1/2, 1/2, -1/2, 0, 0, 0, 0, 1/2), (0, 0, 0, 0, -1, 0, 0, 0),
(0, 1/2, 1/2, 0, -1/2, 1/2, 0, 0),

(-1/2, 1/2, 0, 0, 0, -1/2, -1/2, 0), (0, 1/2, -1/2, 1/2, 0, 0, -1/2, 0),
(-1/2, 1/2, 1/2, 0, 0, 0, 0, 1/2), (0, 0, -1/2, 1/2, -1/2, 0, 0, 1/2),
(-1/2, 1/2, 0, 1/2, -1/2, 0, 0, 0), (1/2, -1/2, 0, -1/2, -1/2, 0, 0, 0),
(1/2, 0, 1/2, -1/2, 0, 1/2, 0, 0), (0, 0, 0, -1/2, 1/2, -1/2, -1/2, 0),
(1/2, 0, -1/2, 0, 1/2, 0, -1/2, 0), (0, 0, 1/2, -1/2, 1/2, 0, 0, 1/2),
(1/2, -1/2, -1/2, 0, 0, 0, 0, 1/2), (1/2, -1/2, 0, 0, 0, -1/2, -1/2, 0),
(1/2, 0, 1/2, 0, 1/2, 0, -1/2, 0), (1/2, -1/2, 1/2, 0, 0, 0, 0, 1/2),
(0, 0, 0, -1/2, -1/2, 1/2, 1/2, 0), (-1/2, 1/2, 0, -1/2, 1/2, 0, 0, 0),
(0, 0, -1, 0, 0, 0, 0, 0), (0, 0, 1/2, -1/2, -1/2, 0, 0, -1/2),
(1/2, 0, 0, 0, -1/2, 1/2, 0, -1/2),
(-1/2, 0, 1/2, -1/2, 0, -1/2, 0, 0), (1/2, 0, -1/2, 1/2, 0, 1/2, 0, 0),
(-1/2, 0, 0, 0, 1/2, -1/2, 0, 1/2), (0, 0, -1/2, 1/2, 1/2, 0, 0, 1/2),
(0, 0, 1, 0, 0, 0, 0, 0),
(1/2, -1/2, 0, 1/2, -1/2, 0, 0, 0), (0, 0, 0, 1/2, 1/2, -1/2, -1/2, 0),
(-1/2, 1/2, -1/2, 0, 0, 0, 0, -1/2), (-1/2, 0, -1/2, 0, -1/2, 0, 1/2, 0),
(-1/2, 1/2, 0, 0, 0, 1/2, 1/2, 0), (-1/2, 1/2, 1/2, 0, 0, 0, 0, -1/2),
(0, 0, -1/2, 1/2, -1/2, 0, 0, -1/2), (-1/2, 0, 1/2, 0, -1/2, 0, 1/2, 0),
(0, 0, 0, 1/2, -1/2, 1/2, 1/2, 0), (-1/2, 0, -1/2, 1/2, 0, -1/2, 0, 0),
(-1/2, 1/2, 0, 1/2, 1/2, 0, 0, 0), (1/2, -1/2, 0, -1/2, 1/2, 0, 0, 0),
(0, 0, 1/2, -1/2, 1/2, 0, 0, -1/2), (1/2, -1/2, -1/2, 0, 0, 0, 0, -1/2),
(0, -1/2, 1/2, -1/2, 0, 0, 1/2, 0), (1/2, -1/2, 0, 0, 0, 1/2, 1/2, 0),

(0, -1/2, -1/2, 0, 1/2, -1/2, 0, 0), (0, 0, 0, 0, 1, 0, 0, 0),

(1/2, -1/2, 1/2, 0, 0, 0, 0, -1/2),

(0, -1/2, 1/2, 0, 1/2, -1/2, 0, 0), (1/2, -1/2, 0, 1/2, 1/2, 0, 0, 0),

(-1/2, 0, 0, -1/2, 0, 0, 1/2, -1/2), (0, 0, -1/2, 0, 0, 1/2, 1/2, -1/2),

(-1/2, 0, -1/2, 0, 1/2, 0, 1/2, 0), (0, -1/2, 0, 0, -1/2, 0, 1/2, -1/2),

(0, 0, 1/2, 0, 0, 1/2, 1/2, -1/2), (-1/2, 0, 0, 0, 1/2, -1/2, 0, -1/2),

(0, 0, -1/2, 1/2, 1/2, 0, 0, -1/2), (-1/2, 0, 1/2, 0, 1/2, 0, 1/2, 0),

(0, -1/2, -1/2, 1/2, 0, 0, 1/2, 0), (-1/2, 0, 0, 1/2, 0, 0, 1/2, -1/2),

(0, -1/2, 0, 0, 1/2, 0, 1/2, -1/2), (0, 0, 0, -1/2, -1/2, 1/2, -1/2, 0),

(-1/2, 0, 0, -1/2, 0, 0, -1/2, 1/2), (0, 0, -1/2, 0, 0, 1/2, -1/2, 1/2),

(0, 0, 0, 0, 0, 0, -1, 0), (0, -1/2, 0, 0, -1/2, 0, -1/2, 1/2),

(0, 0, 1/2, 0, 0, 1/2, -1/2, 1/2),

(-1/2, 0, -1/2, 0, -1/2, 0, -1/2, 0), (-1/2, 1/2, 0, 0, 0, 1/2, -1/2, 0),

(-1/2, 0, 0, 0, -1/2, 1/2, 0, 1/2), (-1/2, 0, 1/2, 0, -1/2, 0, -1/2, 0),

(0, 0, 0, 1/2, -1/2, 1/2, -1/2, 0), (-1/2, 0, 0, 1/2, 0, 0, -1/2, 1/2),

(0, -1/2, 1/2, -1/2, 0, 0, -1/2, 0), (1/2, -1/2, 0, 0, 0, 1/2, -1/2, 0),

(0, -1/2, 0, 0, 1/2, 0, -1/2, 1/2), (-1/2, 0, 1/2, -1/2, 0, 1/2, 0, 0),

(0, -1/2, -1/2, 0, -1/2, 1/2, 0, 0), (-1/2, 0, -1/2, 0, 1/2, 0, -1/2, 0),

(0, -1/2, 1/2, 0, -1/2, 1/2, 0, 0), (-1/2, 0, 1/2, 0, 1/2, 0, -1/2, 0),

(0, -1/2, -1/2, 1/2, 0, 0, -1/2, 0), (-1/2, 0, 0, 0, -1/2, 1/2, 0, -1/2),

(-1, 0, 0, 0, 0, 0, 0, 0), (-1/2, 0, -1/2, 1/2, 0, 1/2, 0, 0)]

```
    ==============
    ==============
    """

    M = Representations_in_Maximal_Order_Octuple(m)
    L = []

    for i in M:
        L.append(Quaternion_Basis_Representation(i))

    return L



def Multiplication_in_Quaternions((A1,A2,B1,B2,C1,C2,D1,D2),(a1,a2,b1,b2,c1,c2,d1,d2)):
    """
    Given two quaternions in the <1,i,j,k> basis, multiplies them. The output
    is an octuple in the <1,i,j,k> basis.

    ==============
```

```
EXAMPLES:

==============

sage: Multiplication_in_Quaternions((1,0,0,0,0,0,0,0),(9,-1/2,8,7,0,4,3,-1/12))

(9, -1/2, 8, 7, 0, 4, 3, -1/12)

==============

==============

sage: Multiplication_in_Quaternions((0,1,0,0,0,0,0,0),(0,1,0,0,0,0,0,0))

(1, 1, 0, 0, 0, 0, 0, 0)

==============

==============

"""

M0 = A1*a1+A2*a2-B1*b1-B2*b2-C1*c1-C2*c2-D1*d1-D2*d2

M1 = A1*a2+A2*a1+A2*a2-B1*b2-B2*b1-B2*b2-C1*c2-C2*c1-C2*c2-D1*d2-D2*d1-D2*d2

M2 = A1*b1+A2*b2+B1*a1+B2*a2+C1*d1+C2*d2-D1*c1-D2*c2

M3 = A1*b2+A2*b1+A2*b2+B1*a2+B2*a1+B2*a2+C1*d2+C2*d1+C2*d2-D1*c2-D2*c1-D2*c2

M4 = A1*c1+A2*c2-B1*d1-B2*d2+C1*a1+C2*a2+D1*b1+D2*b2

M5 = A1*c2+A2*c1+A2*c2-B1*d2-B2*d2-B2*d1+C1*a2+C2*a1+C2*a2+D1*b2+D2*b1+D2*b2

M6 = A1*d1+A2*d2+B1*c1+B2*c2-C1*b1-C2*b2+D1*a1+D2*a2

M7 = A1*d2+A2*d1+A2*d2+B1*c2+B2*c1+B2*c2-C1*b2-C2*b1-C2*b2+D1*a2+D2*a1+D2*a2


M = (M0,M1,M2,M3,M4,M5,M6,M7)
```

```
    return M

##################################
## This section of code is meant##
## to build the necessary       ##
## commands for computing       ##
## Theta(p)--to use the         ##
## language of Dembele.         ##
##################################

def Action_by_Units((a0,a1,a2,a3,a4,a5,a6,a7)):
    """
    Given an arbitrary quaternion element b, this returns
    the set of all b*u, where u is a unit in the maximal order.

    INPUT:
    A= (a0,a1,a2,a3,a4,a5,a6,a7), presumed to be in the <1,i,j,k> basis.
    OUTPUT:
    [A*u, for u a unit in the maximal order]
```

```
=============
EXAMPLES:
(Note: with the exception of multiplication by 0,
all of the sets should have size 120, which is equal
to the number of elements in the maximal order of
norm 1.)
=============

sage: Action_by_Units((0,0,0,0,0,0,0,0))
[(0, 0, 0, 0, 0, 0, 0, 0)]
=============
=============
sage: Action_by_Units((1,0,0,0,0,0,0,0))
[(1/2, 0, 1/2, -1/2, 0, -1/2, 0, 0), (1, 0, 0, 0, 0, 0, 0, 0),
(1/2, 0, 0, 0, 1/2, -1/2, 0, 1/2),
(0, 1/2, 1/2, -1/2, 0, 0, 1/2, 0), (1/2, 0, -1/2, 0, -1/2, 0, 1/2, 0),
(0, 1/2, -1/2, 0, 1/2, -1/2, 0, 0), (1/2, 0, 1/2, 0, -1/2, 0, 1/2, 0),
(0, 1/2, 1/2, 0, 1/2, -1/2, 0, 0), (1/2, 0, -1/2, 1/2, 0, -1/2, 0, 0),
(0, 1/2, 0, 0, -1/2, 0, 1/2, -1/2), (-1/2, 1/2, 0, 0, 0, -1/2, 1/2, 0),
(0, 1/2, -1/2, 1/2, 0, 0, 1/2, 0), (1/2, 0, 0, -1/2, 0, 0, 1/2, -1/2),
(0, 0, 0, -1/2, 1/2, -1/2, 1/2, 0), (1/2, 0, -1/2, 0, 1/2, 0, 1/2, 0),
```

(1/2, 0, 0, 0, 1/2, -1/2, 0, -1/2), (1/2, -1/2, 0, 0, 0, -1/2, 1/2, 0),
(1/2, 0, 1/2, 0, 1/2, 0, 1/2, 0), (0, 0, -1/2, 0, 0, -1/2, 1/2, -1/2),
(0, 1/2, 0, 0, 1/2, 0, 1/2, -1/2), (0, 0, 0, 0, 0, 0, 1, 0),
(0, 0, 1/2, 0, 0, -1/2, 1/2, -1/2),
(1/2, 0, 0, 1/2, 0, 0, 1/2, -1/2), (0, 0, 0, 1/2, 1/2, -1/2, 1/2, 0),
(0, 1/2, 0, 0, -1/2, 0, -1/2, 1/2), (1/2, 0, 0, -1/2, 0, 0, -1/2, 1/2),
(0, 1/2, 1/2, -1/2, 0, 0, -1/2, 0), (1/2, 0, -1/2, 0, -1/2, 0, -1/2, 0),
(0, 0, 1/2, -1/2, -1/2, 0, 0, 1/2), (1/2, 0, 0, 0, -1/2, 1/2, 0, 1/2),
(0, 0, -1/2, 0, 0, -1/2, -1/2, 1/2), (0, 1/2, 0, 0, 1/2, 0, -1/2, 1/2),
(1/2, 0, 1/2, 0, -1/2, 0, -1/2, 0), (0, 0, 1/2, 0, 0, -1/2, -1/2, 1/2),
(1/2, 0, 0, 1/2, 0, 0, -1/2, 1/2), (-1/2, 1/2, 0, -1/2, -1/2, 0, 0, 0),
(0, 1/2, -1/2, 0, -1/2, 1/2, 0, 0), (-1/2, 1/2, -1/2, 0, 0, 0, 0, 1/2),
(0, 0, 0, 0, -1, 0, 0, 0),
(0, 1/2, 1/2, 0, -1/2, 1/2, 0, 0), (-1/2, 1/2, 0, 0, 0, -1/2, -1/2, 0),
(0, 1/2, -1/2, 1/2, 0, 0, -1/2, 0), (-1/2, 1/2, 1/2, 0, 0, 0, 0, 1/2),
(0, 0, -1/2, 1/2, -1/2, 0, 0, 1/2), (-1/2, 1/2, 0, 1/2, -1/2, 0, 0, 0),
(1/2, -1/2, 0, -1/2, -1/2, 0, 0, 0), (1/2, 0, 1/2, -1/2, 0, 1/2, 0, 0),
(0, 0, 0, -1/2, 1/2, -1/2, -1/2, 0), (1/2, 0, -1/2, 0, 1/2, 0, -1/2, 0),
(0, 0, 1/2, -1/2, 1/2, 0, 0, 1/2), (1/2, -1/2, -1/2, 0, 0, 0, 0, 1/2),
(1/2, -1/2, 0, 0, 0, -1/2, -1/2, 0), (1/2, 0, 1/2, 0, 1/2, 0, -1/2, 0),
(1/2, -1/2, 1/2, 0, 0, 0, 0, 1/2), (0, 0, 0, -1/2, -1/2, 1/2, 1/2, 0),

(-1/2, 1/2, 0, -1/2, 1/2, 0, 0, 0), (0, 0, -1, 0, 0, 0, 0, 0),

(0, 0, 1/2, -1/2, -1/2, 0, 0, -1/2),

(1/2, 0, 0, 0, -1/2, 1/2, 0, -1/2), (-1/2, 0, 1/2, -1/2, 0, -1/2, 0, 0),

(1/2, 0, -1/2, 1/2, 0, 1/2, 0, 0), (-1/2, 0, 0, 0, 1/2, -1/2, 0, 1/2),

(0, 0, -1/2, 1/2, 1/2, 0, 0, 1/2), (0, 0, 1, 0, 0, 0, 0, 0),

(1/2, -1/2, 0, 1/2, -1/2, 0, 0, 0),

(0, 0, 0, 1/2, 1/2, -1/2, -1/2, 0), (-1/2, 1/2, -1/2, 0, 0, 0, 0, -1/2),

(-1/2, 0, -1/2, 0, -1/2, 0, 1/2, 0), (-1/2, 1/2, 0, 0, 0, 1/2, 1/2, 0),

(-1/2, 1/2, 1/2, 0, 0, 0, 0, -1/2), (0, 0, -1/2, 1/2, -1/2, 0, 0, -1/2),

(-1/2, 0, 1/2, 0, -1/2, 0, 1/2, 0), (0, 0, 0, 1/2, -1/2, 1/2, 1/2, 0),

(-1/2, 0, -1/2, 1/2, 0, -1/2, 0, 0), (-1/2, 1/2, 0, 1/2, 1/2, 0, 0, 0),

(1/2, -1/2, 0, -1/2, 1/2, 0, 0, 0), (0, 0, 1/2, -1/2, 1/2, 0, 0, -1/2),

(1/2, -1/2, -1/2, 0, 0, 0, 0, -1/2), (0, -1/2, 1/2, -1/2, 0, 0, 1/2, 0),

(1/2, -1/2, 0, 0, 0, 1/2, 1/2, 0), (0, -1/2, -1/2, 0, 1/2, -1/2, 0, 0),

(0, 0, 0, 0, 1, 0, 0, 0),

(1/2, -1/2, 1/2, 0, 0, 0, 0, -1/2), (0, -1/2, 1/2, 0, 1/2, -1/2, 0, 0),

(1/2, -1/2, 0, 1/2, 1/2, 0, 0, 0), (-1/2, 0, 0, -1/2, 0, 0, 1/2, -1/2),

(0, 0, -1/2, 0, 0, 1/2, 1/2, -1/2), (-1/2, 0, -1/2, 0, 1/2, 0, 1/2, 0),

(0, -1/2, 0, 0, -1/2, 0, 1/2, -1/2), (0, 0, 1/2, 0, 0, 1/2, 1/2, -1/2),

(-1/2, 0, 0, 0, 1/2, -1/2, 0, -1/2), (0, 0, -1/2, 1/2, 1/2, 0, 0, -1/2),

(-1/2, 0, 1/2, 0, 1/2, 0, 1/2, 0), (0, -1/2, -1/2, 1/2, 0, 0, 1/2, 0),

(-1/2, 0, 0, 1/2, 0, 0, 1/2, -1/2), (0, -1/2, 0, 0, 1/2, 0, 1/2, -1/2),

(0, 0, 0, -1/2, -1/2, 1/2, -1/2, 0), (-1/2, 0, 0, -1/2, 0, 0, -1/2, 1/2),

(0, 0, -1/2, 0, 0, 1/2, -1/2, 1/2), (0, 0, 0, 0, 0, 0, -1, 0),

(0, -1/2, 0, 0, -1/2, 0, -1/2, 1/2),

(0, 0, 1/2, 0, 0, 1/2, -1/2, 1/2), (-1/2, 0, -1/2, 0, -1/2, 0, -1/2, 0),

(-1/2, 1/2, 0, 0, 0, 1/2, -1/2, 0), (-1/2, 0, 0, 0, -1/2, 1/2, 0, 1/2),

(-1/2, 0, 1/2, 0, -1/2, 0, -1/2, 0), (0, 0, 0, 1/2, -1/2, 1/2, -1/2, 0),

(-1/2, 0, 0, 1/2, 0, 0, -1/2, 1/2), (0, -1/2, 1/2, -1/2, 0, 0, -1/2, 0),

(1/2, -1/2, 0, 0, 0, 1/2, -1/2, 0), (0, -1/2, 0, 0, 1/2, 0, -1/2, 1/2),

(-1/2, 0, 1/2, -1/2, 0, 1/2, 0, 0), (0, -1/2, -1/2, 0, -1/2, 1/2, 0, 0),

(-1/2, 0, -1/2, 0, 1/2, 0, -1/2, 0), (0, -1/2, 1/2, 0, -1/2, 1/2, 0, 0),

(-1/2, 0, 1/2, 0, 1/2, 0, -1/2, 0), (0, -1/2, -1/2, 1/2, 0, 0, -1/2, 0),

(-1/2, 0, 0, 0, -1/2, 1/2, 0, -1/2), (-1, 0, 0, 0, 0, 0, 0, 0),

(-1/2, 0, -1/2, 1/2, 0, 1/2, 0, 0)]

==========

==========

sage: Action_by_Units((2,3,0,-1,4,0,1,-1))

[(1/2, 7/2, 0, -3/2, -1/2, -2, 2, -3), (2, 3, 0, -1, 4, 0, 1, -1),

(-1/2, 7/2, 1, -3, 1, -3/2, 3/2, 2),

(1/2, 3, -3, -3/2, -1, 2, 5/2, -1/2), (5/2, 3/2, -7/2, -3/2, 3/2, -5/2, -1/2, 1/2),

(-1/2, 4, -1/2, -5/2, 0, 1/2, -3, 0), (5/2, 5/2, -3/2, 3/2, 1/2, -3/2, 7/2, 1/2),

(-1/2, 5, 3/2, 1/2, -1, 3/2, 1, 0), (3/2, 7/2, 1, 1/2, 3/2, -3, -2, 1),

(5/2, 3, -3, 2, -1/2, 1/2, -1, -3/2), (0, 7/2, -2, 0, -7/2, -1, -1/2, 3/2),

(3/2, 3, -2, 1/2, 1, 1, -3/2, 7/2), (-1/2, 3/2, -7/2, -1, 2, 0, 0, -7/2),

(-3, 2, -5/2, -3, -1, -3/2, 1/2, -1/2), (-3/2, 3/2, -5/2, -5/2, 7/2, 1/2, -1/2, 3/2),

(-3/2, 7/2, 1, 1, 2, -1/2, -3/2, -3), (-1, 3/2, -1, 0, 1/2, -5, 3/2, 1/2),

(-3/2, 5/2, -1/2, 1/2, 5/2, 3/2, 7/2, 3/2), (-1, 2, -5/2, 1/2, -1/2, -3, -3, -3/2),

(-3/2, 3, -2, 1, 3/2, 7/2, -1, -1/2), (-1, 1, -4, 0, 0, -1, 2, 3),

(-1, 3, -1/2, 7/2, -3/2, -2, 1, -3/2),

(1/2, 5/2, -1/2, 4, 3, 0, 0, 1/2), (-2, 3, 1/2, 2, 0, -3/2, 1/2, 7/2),

(9/2, 2, 1, -2, -3/2, 1/2, 0, 1/2), (3/2, 1/2, 1/2, -5, 1, 0, 1, -3/2),

(3/2, 2, 1, -3/2, -1, 3, 1/2, -7/2), (7/2, 1/2, 1/2, -3/2, 3/2, -3/2, -5/2, -5/2),

(2, 0, -1, -5/2, -5/2, -3/2, 7/2, 0), (7/2, -1/2, -1, -2, 2, 1/2, 5/2, 2),

(1, 1, 3/2, -7/2, -3/2, -3, -2, 1/2), (1/2, 2, 2, -3, 1/2, 7/2, 0, 3/2),

(7/2, 3/2, 5/2, 3/2, 1/2, -1/2, 3/2, -5/2), (1, 2, 7/2, -1/2, -5/2, -2, 2, 1/2),

(5/2, 3/2, 7/2, 0, 2, 0, 1, 5/2), (2, 1/2, -5/2, -2, -7/2, 1/2, -1, -2),

(7/2, 0, -5/2, -3/2, 1, 5/2, -2, 0), (1, 1/2, -3/2, -7/2, -2, 1, -3/2, 3),

(4, 0, -1, 1, -2, -3, 0, -1),

(7/2, 1, -1/2, 3/2, 0, 7/2, 2, 0), (1, 5/2, 2, 0, -7/2, 0, -5/2, -3/2),

(5/2, 2, 2, 1/2, 1, 2, -7/2, 1/2),

(1, 3/2, 1/2, -1/2, -3, 2, 5/2, 3), (3, 0, 0, -1/2, -1/2, -5/2, -1/2, 4),

(3, 3/2, 1/2, 3, -5/2, 1/2, -1, 2), (1, -3/2, -3/2, -2, 1/2, -7/2, 1, -3),

(1/2, -1/2, -1, -3/2, 5/2, 3, 3, -2), (-2, 1, 3/2, -3, -1, -1/2, -3/2, -7/2),

(-1/2, 1/2, 3/2, -5/2, 7/2, 3/2, -5/2, -3/2), (-2, 0, 0, -7/2, -1/2, 3/2, 7/2, 1),

(0, -3/2, -1/2, -7/2, 2, -3, 1/2, 2), (0, 1/2, 3, 0, 1/2, -4, -1/2, -5/2),

(-1/2, 3/2, 7/2, 1/2, 5/2, 5/2, 3/2, -3/2), (0, -1/2, 3/2, -1/2, 1, -2, 9/2, 2),

(1, -2, -9/2, -2, 0, 1/2, 3/2, -1/2), (-2, 1/2, -3/2, -3, -3/2, 7/2, -1, -1),

(0, -1, -2, -3, 1, -1, -4, 0), (1, 0, -1, 3/2, -3/2, -1/2, 1/2, -5),

(5/2, -1/2, -1, 2, 3, 3/2, -1/2, -3),

(-3/2, 1/2, 0, -1/2, -9/2, -2, 1, -2), (3/2, -1/2, 0, 1/2, 9/2, 2, -1, 2),

(-5/2, 1/2, 1, -2, -3, -3/2, 1/2, 3), (-1, 0, 1, -3/2, 3/2, 1/2, -1/2, 5),

(0, 1, 2, 3, -1, 1, 4, 0),

(2, -1/2, 3/2, 3, 3/2, -7/2, 1, 1), (-1, 2, 9/2, 2, 0, -1/2, -3/2, 1/2),

(0, 1/2, -3/2, 1/2, -1, 2, -9/2, -2), (1/2, -3/2, -7/2, -1/2, -5/2, -5/2, -3/2, 3/2),

(0, -1/2, -3, 0, -1/2, 4, 1/2, 5/2), (0, 3/2, 1/2, 7/2, -2, 3, -1/2, -2),

(2, 0, 0, 7/2, 1/2, -3/2, -7/2, -1), (1/2, -1/2, -3/2, 5/2, -7/2, -3/2, 5/2, 3/2),

(2, -1, -3/2, 3, 1, 1/2, 3/2, 7/2), (-1/2, 1/2, 1, 3/2, -5/2, -3, -3, 2),

(-1, 3/2, 3/2, 2, -1/2, 7/2, -1, 3), (-3, -3/2, -1/2, -3, 5/2, -1/2, 1, -2),

(-3, 0, 0, 1/2, 1/2, 5/2, 1/2, -4), (-1, -3/2, -1/2, 1/2, 3, -2, -5/2, -3),

(-5/2, -2, -2, -1/2, -1, -2, 7/2, -1/2), (-1, -5/2, -2, 0, 7/2, 0, 5/2, 3/2),

(-7/2, -1, 1/2, -3/2, 0, -7/2, -2, 0), (-4, 0, 1, -1, 2, 3, 0, 1),

(-1, -1/2, 3/2, 7/2, 2, -1, 3/2, -3),

(-7/2, 0, 5/2, 3/2, -1, -5/2, 2, 0), (-2, -1/2, 5/2, 2, 7/2, -1/2, 1, 2),

```
(-5/2, -3/2, -7/2, 0, -2, 0, -1, -5/2), (-1, -2, -7/2, 1/2, 5/2, 2, -2, -1/2),
(-7/2, -3/2, -5/2, -3/2, -1/2, 1/2, -3/2, 5/2), (-1/2, -2, -2, 3, -1/2, -7/2, 0, -3/2),
(-1, -1, -3/2, 7/2, 3/2, 3, 2, -1/2), (-7/2, 1/2, 1, 2, -2, -1/2, -5/2, -2),
(-2, 0, 1, 5/2, 5/2, 3/2, -7/2, 0), (-7/2, -1/2, -1/2, 3/2, -3/2, 3/2, 5/2, 5/2),
(-3/2, -2, -1, 3/2, 1, -3, -1/2, 7/2), (-3/2, -1/2, -1/2, 5, -1, 0, -1, 3/2),
(-9/2, -2, -1, 2, 3/2, -1/2, 0, -1/2), (2, -3, -1/2, -2, 0, 3/2, -1/2, -7/2),
(-1/2, -5/2, 1/2, -4, -3, 0, 0, -1/2), (1, -3, 1/2, -7/2, 3/2, 2, -1, 3/2),
(1, -1, 4, 0, 0, 1, -2, -3),
(3/2, -3, 2, -1, -3/2, -7/2, 1, 1/2), (1, -2, 5/2, -1/2, 1/2, 3, 3, 3/2),
(3/2, -5/2, 1/2, -1/2, -5/2, -3/2, -7/2, -3/2), (1, -3/2, 1, 0, -1/2, 5, -3/2, -1/2),
(3/2, -7/2, -1, -1, -2, 1/2, 3/2, 3), (3/2, -3/2, 5/2, 5/2, -7/2, -1/2, 1/2, -3/2),
(3, -2, 5/2, 3, 1, 3/2, -1/2, 1/2), (1/2, -3/2, 7/2, 1, -2, 0, 0, 7/2),
(-3/2, -3, 2, -1/2, -1, -1, 3/2, -7/2), (0, -7/2, 2, 0, 7/2, 1, 1/2, -3/2),
(-5/2, -3, 3, -2, 1/2, -1/2, 1, 3/2), (-3/2, -7/2, -1, -1/2, -3/2, 3, 2, -1),
(1/2, -5, -3/2, -1/2, 1, -3/2, -1, 0), (-5/2, -5/2, 3/2, -3/2, -1/2, 3/2, -7/2, -1/2),
(1/2, -4, 1/2, 5/2, 0, -1/2, 3, 0), (-5/2, -3/2, 7/2, 3/2, -3/2, 5/2, 1/2, -1/2),
(-1/2, -3, 3, 3/2, 1, -2, -5/2, 1/2), (1/2, -7/2, -1, 3, -1, 3/2, -3/2, -2),
(-2, -3, 0, 1, -4, 0, -1, 1), (-1/2, -7/2, 0, 3/2, 1/2, 2, -2, 3)]
=============
=============
"""
```

```
    UNITS = Representations_in_Max_Order_Trad_Basis(1)

    MULTS = []
    for i in UNITS:
        if MULTS.count(Multiplication_in_Quaternions(i,(a0,a1,a2,a3,a4,a5,a6,a7))) == 0:
            MULTS.append(Multiplication_in_Quaternions(i,(a0,a1,a2,a3,a4,a5,a6,a7)))
    return MULTS
```



```
def Representatives_up_to_Unit(L):
    """
    INPUT:
    A list of 8-tuples, all corresponding to quaternions in the <1,i,j,k> basis.

    OUTPUT:
    A list of 8-tuples in the <1,i,j,k> corresponding to the original list, with elements
    distinct up to multiplication by a unit in the quaternion order.

    =============
```

```
EXAMPLES:
==========
sage: L = Representations_in_Max_Order_Trad_Basis(1)
sage: Representatives_up_to_Unit(L)
[(1/2, 0, 1/2, -1/2, 0, -1/2, 0, 0)]
==========
==========
sage: J = Representations_in_Max_Order_Trad_Basis(2)
sage: Representatives_up_to_Unit(J)
[(1, 0, 0, -1/2, 1/2, -1/2, 1/2, 0), (1/2, 1/2, 1/2, -1/2, 1/2, -1/2, 1/2, -1/2),
(1, 0, -1/2, 0, 0, -1/2, 1/2, -1/2), (1/2, 0, 1/2, -1/2, 0, -1/2, 1, 0),
(1/2, 1/2, 0, 0, 1, -1/2, 1/2, 0)]
==========
==========
"""
M = []
for k in L:
    M.append(k)
i = 0
while i < len(M):
    UM = Action_by_Units(M[i])
```

```
        for j in UM:
            if M.count(j) >0:
                if M.index(j) != i:
                    M.remove(j)
        i = i+1
    return M



def Representatives_up_to_Unit_e_Basis(L):
    """
    The Representatives_up_to_Unit command assumes
    a <1,i,j,k> basis. This translates that result into
    the e_i basis, where again:

    e_1 = 1/2*(1-(1-a)/2*i+(1+a)/2*j)
    e_2= 1/2*(-(1-a)/2*i+j+(1+a)/2*k)
    e_3 = 1/2*((1+a)/2*i-(1-a)/2*j+k)
    e_4 = 1/2*(i+(1+a)/2*j-(1-a)/2*k).

    =============
    EXAMPLES:
```

```
=============
sage: L = Representations_in_Max_Order_Trad_Basis(1)
sage: Representatives_up_to_Unit(L)
[(1/2, 0, 1/2, -1/2, 0, -1/2, 0, 0)]
sage: Representatives_up_to_Unit_e_Basis(L)
[[           1  1/2*a - 1/2  1/2*a - 1/2 -1/2*a - 3/2]]
=============
=============
sage: J = Representations_in_Max_Order_Trad_Basis(2)
sage: Representatives_up_to_Unit(J)
[(1, 0, 0, -1/2, 1/2, -1/2, 1/2, 0), (1/2, 1/2, 1/2, -1/2, 1/2, -1/2, 1/2, -1/2),
(1, 0, -1/2, 0, 0, -1/2, 1/2, -1/2), (1/2, 0, 1/2, -1/2, 0, -1/2, 1, 0),
(1/2, 1/2, 0, 0, 1, -1/2, 1/2, 0)]
sage: Representatives_up_to_Unit_e_Basis(J)
[[           2            a  1/2*a - 3/2 -1/2*a - 5/2],
[ 1/2*a + 3/2  1/2*a - 1/2  1/2*a - 1/2 -1/2*a - 5/2],
[           2  1/2*a - 1/2  1/2*a - 1/2 -1/2*a - 5/2],
[           1            a  1/2*a - 1/2 -1/2*a - 5/2],
[ 1/2*a + 3/2  1/2*a + 1/2  1/2*a - 1/2 -1/2*a - 5/2]]
=============
=============
```

```
    """
    VEC=[]
    REP=[]
    N = Representatives_up_to_Unit(L)

    MM = Matrix(K,4,4,[2, (1+a)/2-1, (1+a)/2 -1, -1-(1+a)/2,
     0, -1, (1+a)/2, 0,  0,0,-1, (1+a)/2,   0, (1+a)/2, 0, -1])

    for n in N:
        VEC.append(Matrix(K,1,4,[n[0]+(1+a)/2*n[1],
        n[2]+(1+a)/2*n[3], n[4]+(1+a)/2*n[5],
        n[6]+(1+a)/2*n[7]]))
    for v in VEC:
        REP.append(v*MM)
    return REP


def Theta_p(m):
    """
    Let m be a totally positive element of O_K. In particular
    suppose m is the totally positive generator of a prime ideal
```

```
in O_K.

This returns (with respect to the e_i basis) the set of
elements in the maximal order with norm m
UP TO EQUIVALENCE BY A UNIT IN THE ORDER.

=============
EXAMPLES:
=============
sage: Theta_p(2)
[[           2            a  1/2*a - 3/2 -1/2*a - 5/2],
[ 1/2*a + 3/2  1/2*a - 1/2  1/2*a - 1/2 -1/2*a - 5/2],
[           2  1/2*a - 1/2  1/2*a - 1/2 -1/2*a - 5/2],
[           1            a  1/2*a - 1/2 -1/2*a - 5/2],
[ 1/2*a + 3/2  1/2*a + 1/2  1/2*a - 1/2 -1/2*a - 5/2]]
=============
=============
sage: Theta_p(1/2*a+5/2)
[[          2           a 1/2*a - 1/2      -a - 3],
[1/2*a + 5/2           a 1/2*a - 1/2      -a - 3],
[1/2*a + 3/2 1/2*a + 1/2 1/2*a - 1/2      -a - 3],
```

```
    [1/2*a + 5/2 1/2*a + 1/2 1/2*a - 1/2      -a - 3],
    [1/2*a + 3/2       a + 1 1/2*a - 1/2      -a - 3],
    [          2       a + 1 1/2*a - 1/2      -a - 3]]
    =============
    =============
    """
    assert (K.ideal(m)).is_prime() == True
    assert (K(m)).is_totally_positive() == True

    theta = Representations_in_Max_Order_Trad_Basis(m)
    Theta = Representatives_up_to_Unit_e_Basis(theta)

    return Theta

##################################
## This section of code is meant##
## to build necessary commands  ##
## for element manipulation in  ##
## O_K/ I*O_K for I an ideal.   ##
##################################
```

```
def List_of_Residues(I):
    """
    Given an ideal I of O_K, this returns a set of representatives of O_K/I*O_K.

    =============
    EXAMPLES:
    =============
    =============
    sage: List_of_Residues(K.ideal(2))
    [0, a, 1/2*a + 1/2, 3/2*a + 1/2]
    =============
    =============
    sage: List_of_Residues(K.ideal(a))
    [-a - 1, -1/2*a - 1/2, 0, 1/2*a + 1/2, a + 1]
    =============
    =============
    sage: List_of_Residues(K.ideal(6))
    [-3*a - 1, -2*a - 1, -a - 1, -1, a - 1, 2*a - 1, -5/2*a - 1/2, -3/2*a - 1/2,
    -1/2*a - 1/2, 1/2*a - 1/2, 3/2*a - 1/2, 5/2*a - 1/2, -2*a, -a, 0, a,
    2*a, 3*a, -3/2*a + 1/2, -1/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2,
    5/2*a + 1/2, 7/2*a + 1/2, -a + 1, 1, a + 1, 2*a + 1, 3*a + 1, 4*a + 1,
```

```
    -1/2*a + 3/2, 1/2*a + 3/2, 3/2*a + 3/2, 5/2*a + 3/2, 7/2*a + 3/2, 9/2*a + 3/2]
    ==============
    ==============
    """
    x = QQ['x'].0
    K.<a> = NumberField(x^2-5,embedding=1)
    OK = K.ring_of_integers()

    assert I.is_integral() == True
    assert I != K.ideal(0)

    L=[]
    R = I.residues()
    for i in R:
        L.append(i)
    return L



def Unit_Residues(I):
```

```
"""
Given a nonzero ideal I, returns the representatives of invertible elements.
N.B. For a prime ideal, this will return all residues save 0.

=============
EXAMPLES:
=============
=============
sage: Unit_Residues(K.ideal(2))
[a, 1/2*a + 1/2, 3/2*a + 1/2]
=============
=============
sage: Unit_Residues(K.ideal(a))
[-a - 1, -1/2*a - 1/2, 1/2*a + 1/2, a + 1]
=============
=============
sage: Unit_Residues(K.ideal(6))
[-2*a - 1, -1, 2*a - 1, -5/2*a - 1/2, -3/2*a - 1/2, -1/2*a - 1/2,
1/2*a - 1/2, 3/2*a - 1/2, 5/2*a - 1/2, -a, a, -3/2*a + 1/2,
-1/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2, 5/2*a + 1/2, 7/2*a + 1/2,
1, 2*a + 1, 4*a + 1, -1/2*a + 3/2, 1/2*a + 3/2, 5/2*a + 3/2, 7/2*a + 3/2]
```

```
    ==============
    ==============
    """
    x = QQ['x'].0
    K.<a> = NumberField(x^2-5,embedding=1)
    OK = K.ring_of_integers()

    assert I.is_integral() == True
    assert I != K.ideal(0)

    L=[]
    R = I.residues()
    for i in R:
        if gcd(i.norm(), I.norm()) == 1:
            L.append(i)
    return L



def Residue_mod_Ideal(m,I):
```

```
"""
Considering an integer m in O_K and an
integral ideal I, this code returns
a in O_K/I*O_K satisfying
a \equiv m mod(I).

=============
EXAMPLES:
=============
=============
sage: Residue_mod_Ideal(1, K.ideal(2))
a
=============
=============
sage: Residue_mod_Ideal(5, K.ideal(a))
0
=============
=============
sage: Residue_mod_Ideal(1/2, K.ideal(2))
Traceback (most recent call last):
...
```

```
AssertionError

=============

=============

"""

x = QQ['x'].0

K.<a> = NumberField(x^2-5,embedding=1)

OK = K.ring_of_integers()


assert K(m).is_integral() == True

assert I.is_integral() == True

assert I != K.ideal(0)


Res = List_of_Residues(I)


for i in Res:

    if m-i == 0:

        return i

    if I.divides(K.ideal(m-i)):

        return i
```

```
def Inverse_mod_Ideal(m,I):
    """
    This will take an ideal I, a value m
    which is invertible in I, and will
    return its inverse in O_K/I*O_K.
    =============
    EXAMPLES:
    =============
    =============
    sage: Inverse_mod_Ideal(2, K.ideal(2))
    Traceback (most recent call last):
    ...
    AssertionError
    =============
    =============
    sage: Inverse_mod_Ideal(a+1, K.ideal(3))
    -1/2*a + 1/2
    =============
    =============
```

```
sage: Inverse_mod_Ideal(6, K.ideal(a))
a + 1
==============
==============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)
N = I.norm()

assert gcd(N, (K.ideal(m)).norm()) == 1

Res = List_of_Residues(I)

for i in Res:
    if gcd(i.norm(), N) == 1:
        if gcd(N, (i*m-1).norm()) > 0:
            if Residue_mod_Ideal(i*m -1 , I) == 0:
```

```
                    return i



###################################
## This section of code is meant##
## to build the necessary       ##
## commands for projective      ##
## spaces P^1(O_K/I*O_K)        ##
## fixed trivial                ##
## character                    ##
###################################



def Projective_Space_Prime_Powers(I):
    """
    Dembele claims that for O_F/p^e representatives of P^1 are
    (1,a) for a in p/p^e
    (a,1) for a in O_F/p^e
```

```
We will return a list of all such values.

=============

EXAMPLES:

=============

sage: Projective_Space_Prime_Powers(K.ideal(2))

[(1, 0), (0, a), (a, a), (1/2*a + 1/2, a), (3/2*a + 1/2, a)]

=============

sage: Projective_Space_Prime_Powers(K.ideal(8))

[(1, 0), (-9/2*a - 3/2, 1), (-7/2*a - 3/2, 1), (-5/2*a - 3/2, 1), (-3/2*a - 3/2, 1),

(-1/2*a - 3/2, 1), (1/2*a - 3/2, 1), (3/2*a - 3/2, 1), (5/2*a - 3/2, 1),

(-4*a - 1, 1), (-3*a - 1, 1), (-2*a - 1, 1), (-a - 1, 1), (-1, 1),

(a - 1, 1), (2*a - 1, 1), (3*a - 1, 1), (-7/2*a - 1/2, 1), (-5/2*a - 1/2, 1),

(-3/2*a - 1/2, 1), (-1/2*a - 1/2, 1), (1/2*a - 1/2, 1), (3/2*a - 1/2, 1),

(5/2*a - 1/2, 1), (7/2*a - 1/2, 1), (-3*a, 1), (-2*a, 1), (-a, 1), (0, 1),

(a, 1), (2*a, 1), (3*a, 1), (4*a, 1), (-5/2*a + 1/2, 1), (-3/2*a + 1/2, 1),

(-1/2*a + 1/2, 1), (1/2*a + 1/2, 1), (3/2*a + 1/2, 1), (5/2*a + 1/2, 1),

(7/2*a + 1/2, 1), (9/2*a + 1/2, 1), (-2*a + 1, 1), (-a + 1, 1), (1, 1),

(a + 1, 1), (2*a + 1, 1), (3*a + 1, 1), (4*a + 1, 1), (5*a + 1, 1),

(-3/2*a + 3/2, 1), (-1/2*a + 3/2, 1), (1/2*a + 3/2, 1), (3/2*a + 3/2, 1),

(5/2*a + 3/2, 1), (7/2*a + 3/2, 1), (9/2*a + 3/2, 1), (11/2*a + 3/2, 1),

(-a + 2, 1), (2, 1), (a + 2, 1), (2*a + 2, 1), (3*a + 2, 1), (4*a + 2, 1),
```

```
(5*a + 2, 1), (6*a + 2, 1), (1, -3*a - 1), (1, -a - 1), (1, a - 1), (1, 3*a - 1),
(1, -2*a), (1, 2*a), (1, 4*a), (1, -a + 1), (1, a + 1), (1, 3*a + 1), (1, 5*a + 1),
(1, 2), (1, 2*a + 2), (1, 4*a + 2), (1, 6*a + 2)]
=============
sage: Projective_Space_Prime_Powers(K.ideal(2*a))
Traceback (most recent call last):
...
AssertionError
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)

assert len(I.factor()) == 1

Proj=[(1,0)]
```

```
    LoRes = List_of_Residues(I)
    for a in LoRes:
        Proj.append((a,Residue_mod_Ideal(1, I)))
    for a in LoRes:
        if a != 0:
            if (I.factor()[0][0]).divides(K.ideal(a)):
                Proj.append((Residue_mod_Ideal(1,I),a))
    return Proj
```



```
def Projective_Space_Arbitrary(I):
    """
    This uses the Chinese Remainder Theorem on
    the above function.
    =============
    EXAMPLES:
    =============
    sage: Projective_Space_Arbitrary(K.ideal(2))
    [(1, 0), (0, a), (a, a), (1/2*a + 1/2, a), (3/2*a + 1/2, a)]
```

```
=============
sage: Projective_Space_Arbitrary(K.ideal(3))
[(1, 0), (-3/2*a - 1/2, -3/2*a - 1/2), (-1/2*a - 1/2, -3/2*a - 1/2),
(1/2*a - 1/2, -3/2*a - 1/2), (-a, -3/2*a - 1/2), (0, -3/2*a - 1/2),
(a, -3/2*a - 1/2), (-1/2*a + 1/2, -3/2*a - 1/2),
(1/2*a + 1/2, -3/2*a - 1/2), (3/2*a + 1/2, -3/2*a - 1/2)]
=============
sage: Projective_Space_Arbitrary(K.ideal(6))
[(1, 0), (1, 3*a + 1), (4*a + 1, 3*a + 1), (2*a + 1, 3*a + 1),
(-a, 3*a + 1), (3*a, 3*a + 1), (a, 3*a + 1), (-2*a - 1, 3*a + 1),
(2*a - 1, 3*a + 1), (-1, 3*a + 1), (3*a + 1, 3*a), (3*a + 1, 1),
(a + 1, 1), (-a + 1, 1), (2*a, 1), (0, 1), (-2*a, 1), (a - 1, 1),
(-a - 1, 1), (-3*a - 1, 1), (1, 3*a), (1, 1), (4*a + 1, 1), (2*a + 1, 1),
(-a, 1), (3*a, 1), (a, 1), (-2*a - 1, 1), (2*a - 1, 1), (-1, 1),
(3/2*a - 1/2, 3*a), (3/2*a - 1/2, 1), (-1/2*a - 1/2, 1), (-5/2*a - 1/2, 1),
(7/2*a + 3/2, 1), (3/2*a + 3/2, 1), (-1/2*a + 3/2, 1), (5/2*a + 1/2, 1),
(1/2*a + 1/2, 1), (-3/2*a + 1/2, 1), (-3/2*a - 1/2, 3*a), (-3/2*a - 1/2, 1),
(5/2*a - 1/2, 1), (1/2*a - 1/2, 1), (1/2*a + 3/2, 1), (9/2*a + 3/2, 1),
(5/2*a + 3/2, 1), (-1/2*a + 1/2, 1), (7/2*a + 1/2, 1), (3/2*a + 1/2, 1)]

=============
```

```
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)

FACTOR = I.factor()
PRIMEPOWERPROJECTIVES=[]
INVERSES=[]
PAIRSOPAIRS=[]
TOTAL=[]
if len(I.factor()) == 1:
    return Projective_Space_Prime_Powers(I)
else:
    for f in FACTOR:

        PRIMEPOWERPROJECTIVES.append(Projective_Space_Prime_Powers(f[0]^(f[1])))

        INVERSES.append((Inverse_mod_Ideal((I/(f[0]^(f[1]))).gens_reduced()[0],
```

```
            (f[0]^(f[1]))), (I/(f[0]^(f[1]))).gens_reduced()[0]))

        N = cartesian_product_iterator(PRIMEPOWERPROJECTIVES)
        for n in N:
            PAIRSOPAIRS.append(n)
        for i in range(len(PAIRSOPAIRS)):
            TOTAL.append((  Residue_mod_Ideal(sum(INVERSES[j][0]*INVERSES[j][1]*PAIRSOPAIRS[i][j][0]
            for j in range(len(PAIRSOPAIRS[i]))),I),
            Residue_mod_Ideal(sum(INVERSES[j][0]*INVERSES[j][1]*PAIRSOPAIRS[i][j][1]
            for j in range(len(PAIRSOPAIRS[i]))),I)   ))
        return TOTAL
```



```
def Projective_Space_Representative((A,B),I):
    """
    Need to take a tuple (A,B) and write it as a representative in Projective Space
    M=(a,b) of P^1(O_K/I*O_K)

    Note that (A,B) cong (C,D) iff there is a unit u of I with A*u=C and B*u=D.
```

```
=============
EXAMPLES:
=============
sage: Projective_Space_Representative((1,1), K.ideal(2))
(a, a)
=============
sage: Projective_Space_Representative((1,1), K.ideal(3))
(-3/2*a - 1/2, -3/2*a - 1/2)
=============
sage: Projective_Space_Representative((1,1), K.ideal(6))
(1, 1)
=============
sage: Projective_Space_Representative((0,0), K.ideal(6))
Traceback (most recent call last):
...
AssertionError
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
```

```
    OK = K.ring_of_integers()

    assert (Residue_mod_Ideal(A,I), Residue_mod_Ideal(B,I)) !=
     (Residue_mod_Ideal(0,I),Residue_mod_Ideal(0,I))
    REPS= Projective_Space_Arbitrary(I)

    M = (Residue_mod_Ideal(A,I), Residue_mod_Ideal(B,I) )

    N=[]

    for r in REPS:
        if A*r[1] - B*r[0] == 0:
            return r
        else:
            if I.divides(K.ideal(A*r[1] - B*r[0])):
                return r



##################################
```

```
## This section of code is meant##
## to build the necessary       ##
## commands for local           ##
## isomorphisms at split        ##
## primes                       ##
##################################


def Local_Isomorphism_for_Odd_Prime_Ideal_Trad(I):
    """
    This assumes we have an odd prime ideal, I.
    This returns where to send <1,i,j,k>. This uses
    Lemma 5.1 of Voight's paper.

    We will ensure that v is not congruent to 0 mod p, for
    reasons which will be apparent later.

    =============
    EXAMPLES:
    =============
    =============
```

```
sage: Local_Isomorphism_for_Odd_Prime_Ideal_Trad(K.ideal(2))
Traceback (most recent call last):
...
AssertionError
=============
=============
sage: Local_Isomorphism_for_Odd_Prime_Ideal_Trad(K.ideal(3))
([-3/2*a - 1/2, 0, 0, -3/2*a - 1/2], [0, 3/2*a + 1/2, -3/2*a - 1/2, 0],
[-3/2*a - 1/2, -3/2*a - 1/2, -3/2*a - 1/2, 3/2*a + 1/2],
[3/2*a + 1/2, -3/2*a - 1/2, -3/2*a - 1/2, -3/2*a - 1/2])
=============
=============
sage: Local_Isomorphism_for_Odd_Prime_Ideal_Trad(K.ideal(a))
([a + 1, 0, 0, a + 1], [0, -a - 1, a + 1, 0],
[0, -1/2*a - 1/2, -1/2*a - 1/2, 0], [1/2*a + 1/2, 0, 0, -1/2*a - 1/2])
=============
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()
```

```
assert I != K.ideal(0)


assert I.norm() %2 == 1 %2
assert I.is_prime() == True


Res = List_of_Residues(I)
U = Unit_Residues(I)
NegOneSet = []


for u in Res:
    for v in U:
        if Residue_mod_Ideal(u^2+v^2,I) == Residue_mod_Ideal(-1,I):
            NegOneSet.append((u,v))
U = NegOneSet[0][0]
V = NegOneSet[0][1]



ID = [Residue_mod_Ideal(1,I), Residue_mod_Ideal(0,I), Residue_mod_Ideal(0,I), Residue_mod_Ideal(1,I)]
iMap = [Residue_mod_Ideal(0,I), Residue_mod_Ideal(-1,I), Residue_mod_Ideal(1,I), Residue_mod_Ideal(0,I)]
```

```
    jMap = [Residue_mod_Ideal(U,I), Residue_mod_Ideal(V,I), Residue_mod_Ideal(V,I), Residue_mod_Ideal(-U,I)]
    kMap = [Residue_mod_Ideal(-V,I), Residue_mod_Ideal(U,I), Residue_mod_Ideal(U,I), Residue_mod_Ideal(V,I)]
    return [ID, iMap, jMap, kMap]



def Local_Isomorphism_Powers_of_Odd_Prime_Trad(I):
    """
    This code differs from Local_Isomorphism_for_Odd_Prime_Ideal(I)
    in that for ideals of the form I=p^e for e>1, a Hensel lift
    is performed on the jMap (and subsequently the k-map).
    Again, this only holds for the <1,i,j,k> basis

    =============
    EXAMPLES:
    =============
    =============
    sage: Local_Isomorphism_Powers_of_Odd_Prime_Trad(K.ideal(a^3))
    ([1, 0, 0, 1], [0, -1, 1, 0],
    [5*a + 5, -15/2*a - 11/2, -15/2*a - 11/2, -5*a - 5],
    [15/2*a + 11/2, 5*a + 5, 5*a + 5, -15/2*a - 11/2])
```

```
=============
=============
sage: Local_Isomorphism_Powers_of_Odd_Prime_Trad(K.ideal(5))
([1, 0, 0, 1], [0, -1, 1, 0],
[0, -5/2*a - 1/2, -5/2*a - 1/2, 0],
[5/2*a + 1/2, 0, 0, -5/2*a - 1/2])
=============
=============
sage: Local_Isomorphism_Powers_of_Odd_Prime_Trad(K.ideal(a))
([a + 1, 0, 0, a + 1], [0, -a - 1, a + 1, 0],
[0, -1/2*a - 1/2, -1/2*a - 1/2, 0],
[1/2*a + 1/2, 0, 0, -1/2*a - 1/2])
=============
=============
sage: Local_Isomorphism_Powers_of_Odd_Prime_Trad(K.ideal(3))
([-3/2*a - 1/2, 0, 0, -3/2*a - 1/2],
[0, 3/2*a + 1/2, -3/2*a - 1/2, 0],
[-3/2*a - 1/2, -3/2*a - 1/2, -3/2*a - 1/2, 3/2*a + 1/2],
[3/2*a + 1/2, -3/2*a - 1/2, -3/2*a - 1/2, -3/2*a - 1/2])
=============
=============
```

```
sage: Local_Isomorphism_for_Odd_Prime_Ideal_Trad(K.ideal(2))
Traceback (most recent call last):
...
AssertionError
=============
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I != K.ideal(0)

assert I.norm() %2 == 1 %2

assert len(I.factor()) == 1
N=I.norm()

if I.is_prime() == True:
    return Local_Isomorphism_for_Odd_Prime_Ideal_Trad(I)
```

```
else:
    ID = [Residue_mod_Ideal(1,I), Residue_mod_Ideal(0,I),
     Residue_mod_Ideal(0,I), Residue_mod_Ideal(1,I)]
    iMap = [Residue_mod_Ideal(0,I), Residue_mod_Ideal(-1,I),
     Residue_mod_Ideal(1,I), Residue_mod_Ideal(0,I)]
    p = I.prime_factors()[0]
    P = p.gens_two()[0]
    e=1
    u = Local_Isomorphism_for_Odd_Prime_Ideal_Trad(p)[2][0]
    v = Local_Isomorphism_for_Odd_Prime_Ideal_Trad(p)[2][1]

    while (K.ideal(p^e)).norm() < N:
        Q = u^2+v^2+1
        T = Inverse_mod_Ideal(2,K.ideal(p^(e+1)))
        V = Inverse_mod_Ideal(v, K.ideal(p^(e+1)))
        A = P^e
        B = -V*(Q*T + A*u)
        e=e+1
        u= u+A
        v= v+B
    jMap = [Residue_mod_Ideal(u,I), Residue_mod_Ideal(v,I),
```

```
         Residue_mod_Ideal(v,I), Residue_mod_Ideal(-u,I)]
        kMap = [Residue_mod_Ideal(-v,I), Residue_mod_Ideal(u,I),
         Residue_mod_Ideal(u,I), Residue_mod_Ideal(v,I)]

        return [ID, iMap, jMap, kMap]
```



```
def Local_Isomorphism_for_Powers_of_Odd_Prime_Ideal(I):
    """
    Takes the _Trad(I) code, and rewrites for e_1,e_2,e_3,e_4.
    e1 = 1/2*(1-(1-\theta)*i+ \theta*j)
    e2 = 1/2*(-(1-\theta)*i+j+\theta *k)
    e3 = 1/2*(\theta *i - (1- \theta)*j +k)
    e4 = 1/2*(i+\theta*j - (1- \theta)*k)
```

```
=============
EXAMPLES:
=============
=============

sage: Local_Isomorphism_for_Powers_of_Odd_Prime_Ideal(K.ideal(2))
Traceback (most recent call last):
...
AssertionError
=============
sage: Local_Isomorphism_for_Powers_of_Odd_Prime_Ideal(K.ideal(15))
Traceback (most recent call last):
...
AssertionError
=============
sage: Local_Isomorphism_for_Powers_of_Odd_Prime_Ideal(K.ideal(5))
[[5/2*a + 1/2, 3/2*a - 1/2, -3*a - 1, 5/2*a + 1/2],
[-1/2*a - 1/2, a, 3/2*a - 1/2, 1/2*a + 1/2],
[-1, 3/2*a + 1/2, 2*a + 1, 1], [-1/2*a + 1/2, -2*a, 3*a + 1, 1/2*a - 1/2]]
=============
sage: Local_Isomorphism_for_Powers_of_Odd_Prime_Ideal(K.ideal(3))
```

```
[[a, 3/2*a + 1/2, -a, 1/2*a - 1/2],
[1/2*a - 1/2, -3/2*a - 1/2, 1/2*a + 1/2, -1/2*a + 1/2],
[a, 0, 1/2*a + 1/2, -a], [3/2*a + 1/2, 1/2*a - 1/2, 1/2*a + 1/2, -3/2*a - 1/2]]
==============
sage: Local_Isomorphism_for_Powers_of_Odd_Prime_Ideal(K.ideal(5*a))
[[-5/2*a - 9/2, 4*a + 2, -1/2*a + 3/2, 15/2*a + 11/2],
[-11/2*a - 11/2, 6*a + 5, 13/2*a + 9/2, 11/2*a + 11/2],
[5/2*a + 3/2, 3/2*a + 1/2, 2*a + 1, -5/2*a - 3/2], [2*a + 3, -2*a, 3*a + 1, -2*a - 3]]
==============

"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()


assert I != K.ideal(0)

assert I.norm() %2 == 1 %2

assert len(I.factor()) == 1
```

```
TB = Local_Isomorphism_Powers_of_Odd_Prime_Trad(I)


ONE = TB[0]

EYE = TB[1]

J = TB[2]

K = TB[3]


W = Inverse_mod_Ideal(2,I)

T = (1+a)/2

E10 = Residue_mod_Ideal(W*(ONE[0]-(1-T)*EYE[0] + T*J[0]  ), I)

E11 = Residue_mod_Ideal(W*(ONE[1]-(1-T)*EYE[1] + T*J[1]  ), I)

E12 = Residue_mod_Ideal(W*(ONE[2]-(1-T)*EYE[2] + T*J[2]  ), I)

E13 = Residue_mod_Ideal(W*(ONE[3]-(1-T)*EYE[3] + T*J[3]  ), I)



E20 = Residue_mod_Ideal(W*(-(1-T)*EYE[0]+J[0]+T*K[0]), I)

E21 = Residue_mod_Ideal(W*(-(1-T)*EYE[1]+J[1]+T*K[1]), I)

E22 = Residue_mod_Ideal(W*(-(1-T)*EYE[2]+J[2]+T*K[2]), I)

E23 = Residue_mod_Ideal(W*(-(1-T)*EYE[3]+J[3]+T*K[3]), I)
```

```
E30 = Residue_mod_Ideal(W*(T*EYE[0]-(1-T)*J[0]+K[0]), I)
E31 = Residue_mod_Ideal(W*(T*EYE[1]-(1-T)*J[1]+K[1]), I)
E32 = Residue_mod_Ideal(W*(T*EYE[2]-(1-T)*J[2]+K[2]), I)
E33 = Residue_mod_Ideal(W*(T*EYE[3]-(1-T)*J[3]+K[3]), I)

E40 = Residue_mod_Ideal(W*(EYE[0]+T*J[0]-(1-T)*K[0]),I)
E41 = Residue_mod_Ideal(W*(EYE[1]+T*J[1]-(1-T)*K[1]),I)
E42 = Residue_mod_Ideal(W*(EYE[2]+T*J[2]-(1-T)*K[2]),I)
E43 = Residue_mod_Ideal(W*(EYE[3]+T*J[3]-(1-T)*K[3]),I)

E1 = [E10, E11,E12,E13]
E2 = [E20,E21,E22,E23]
E3 = [E30,E31,E32,E33]
E4 = [E40,E41,E42,E43]


return [E1,E2,E3,E4]
```

```
def Local_Isomorphism_for_Even_Prime_Ideal(I):
    """
    This gives a local isomorphism
    R \otimes (O_K/2) \equiv M_2(O_K/2).

    Instead of searching for an isomorphism in a way
    similar to the case of odd primes, this code
    is based on an explicit computation.

    =============
    EXAMPLES:
    =============
    =============
    sage: I= K.ideal(2)
    sage: Local_Isomorphism_at_Two(I)
    ((1/2*a + 1/2, a, 0, 3/2*a + 1/2), (a, 1/2*a + 1/2, 0, a),
    (0, a, a, 0), (3/2*a + 1/2, 3/2*a + 1/2, a, 3/2*a + 1/2))
    =============
    =============
```

```
sage: Local_Isomorphism_at_Two(K.ideal(3))
Traceback (most recent call last):
...
AssertionError
=============
=============
sage: Local_Isomorphism_at_Two(K.ideal(4))
Traceback (most recent call last):
...
AssertionError
=============
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I == K.ideal(2)

E1 =(Residue_mod_Ideal((1+a)/2,I),Residue_mod_Ideal(1,I),
Residue_mod_Ideal(0,I),Residue_mod_Ideal(1+(1+a)/2,I))
```

```
        E2 =(Residue_mod_Ideal(1,I),Residue_mod_Ideal((1+a)/2,I),
        Residue_mod_Ideal(0,I),Residue_mod_Ideal(1,I))
        E3 =(Residue_mod_Ideal(0,I),Residue_mod_Ideal(1,I),
        Residue_mod_Ideal(1,I),Residue_mod_Ideal(0,I))
        E4 =(Residue_mod_Ideal(1+(1+a)/2,I),Residue_mod_Ideal(1+(1+a)/2,I),
        Residue_mod_Ideal(1,I),Residue_mod_Ideal(1+(1+a)/2,I))

        return [E1, E2, E3, E4]


    def Hensel_at_Two(I):
        """
        Need to do a Hensel lift at 2 of the vector x used in Local_Isomorphism_at_Two(I).
        Note that mod(2) x=((1+a)/2, 1, 0, 0).

        =============
        EXAMPLES:
        =============
        =============
        sage: Hensel_at_Two(K.ideal(2))
        (1/2*a + 1/2, a, 0, 0)
```

```
=============
sage: Hensel_at_Two(K.ideal(4))
(3/2*a - 1/2, 1, 0, 0)
=============
sage: Hensel_at_Two(K.ideal(32))
(-5/2*a - 1/2, 1, 0, 0)
=============
sage: Hensel_at_Two(K.ideal(3))
Traceback (most recent call last):
...
AssertionError
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I != K.ideal(0)
assert I.norm() %2 == 0 %2
assert len(I.factor()) == 1
N=I.norm()
```

```
if I.is_prime() == True:
    u = (1+a)/2
    v = 1
    return (Residue_mod_Ideal(u, I), Residue_mod_Ideal(v,I),
    Residue_mod_Ideal(0, I), Residue_mod_Ideal(0,I))


else:
    p = I.prime_factors()[0]
    P = p.gens_two()[0]
    e=1
    u = (1+a)/2
    v = 1
    #q = u^2+v^2+u*v
    while (K.ideal(p^e)).norm() < N:
        Q = u^2+v^2+u*v
        V = Inverse_mod_Ideal(2*u+v, K.ideal(p^(e+1)))
        B = 0
        A = -Q*V
        e=e+1
        u= u+A
```

```
            v= v+B
        return (Residue_mod_Ideal(u, I), Residue_mod_Ideal(v,I),
        Residue_mod_Ideal(0, I), Residue_mod_Ideal(0,I))



def Local_Isomorphism_Powers_of_Even_Prime(I):
    """
    We take the Hensel at 2 code and use it to give maps to e_1,e_2,e_3,e_4.
    x = (A,1,0,0), y = (0, (-1+\theta)/A, A*(1- \theta), A+1)
    e_1 \mapsto [A, A-1+ \theta, 1 + \theta + A*\theta, 1-A]
    e_2 \mapsto [A-1/A, 2A-A*\theta +1, \theta -A, 1/A - A]
    e_3 \mapsto [-1-\theta/A, 2\theta + \theta*A -A, 1, 1+ \theta/A]
    e_4 \mapsto [theta*A, A+1, \theta +A+1, - \theta*A]

    =============
    EXAMPLES:
    =============
    =============
    sage: Local_Isomorphism_Powers_of_Even_Prime(K.ideal(2))
```

```
[[1/2*a + 1/2, a, 0, 3/2*a + 1/2], [a, 1/2*a + 1/2, 0, a],
[0, a, a, 0], [3/2*a + 1/2, 3/2*a + 1/2, a, 3/2*a + 1/2]]
=============
sage: Local_Isomorphism_Powers_of_Even_Prime(K.ideal(4))
[[3/2*a - 1/2, 1, a + 1, 1/2*a - 1/2], [-a, 5/2*a + 1/2, 3*a + 1, a],
[3*a + 1, 1, 1, 3*a + 1], [1/2*a - 1/2, 3/2*a + 1/2, 2*a + 1, -1/2*a + 1/2]]
=============
sage: Local_Isomorphism_Powers_of_Even_Prime(K.ideal(16))
[[-5/2*a - 1/2, -2*a - 1, 7*a + 3, 5/2*a + 3/2], [-5*a, 9/2*a - 3/2, 3*a + 1, 5*a],
[7*a + 1, 10*a + 3, 1, -7*a - 1], [13/2*a + 3/2, -5/2*a + 1/2, -2*a + 1, -13/2*a - 3/2]]
=============
sage: Local_Isomorphism_Powers_of_Even_Prime(K.ideal(3))
Traceback (most recent call last):
...
AssertionError
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()
```

```
assert I != K.ideal(0)
assert I.norm() %2 == 0 %2
assert len(I.factor()) == 1
A = Hensel_at_Two(I)[0]
T = Inverse_mod_Ideal(A, I)
S = (1+a)/2

E1 = [A, Residue_mod_Ideal(A-1+S, I), Residue_mod_Ideal(1+S+A*S, I),
Residue_mod_Ideal(1-A,I)]
E2 = [Residue_mod_Ideal(A-T, I), Residue_mod_Ideal(2*A-A*S+1,I),
Residue_mod_Ideal(S-A, I), Residue_mod_Ideal(T-A, I)]
E3 = [Residue_mod_Ideal(-1-S*T, I), Residue_mod_Ideal(2*S+S*A-A, I),
Residue_mod_Ideal(1, I), Residue_mod_Ideal(1+S*T, I)]
E4 = [Residue_mod_Ideal(S*A, I), Residue_mod_Ideal(A+1, I),
Residue_mod_Ideal(S+A+1, I), Residue_mod_Ideal(-S*A, I)]


return [E1, E2, E3, E4]
```

```
def Local_Information_Arbitrary_Ideal(I):
    """
    This combines the even and odd functions to give a divisor-by-divisor
    analysis of the local isomorphisms (i.e., literally Step 3 of Lassina)

    =============
    EXAMPLES:
    =============
    =============
    sage: Local_Information_Arbitrary_Ideal(K.ideal(2))
    [(Fractional ideal (2), [[1/2*a + 1/2, a, 0, 3/2*a + 1/2],
    [a, 1/2*a + 1/2, 0, a], [0, a, a, 0],
    [3/2*a + 1/2, 3/2*a + 1/2, a, 3/2*a + 1/2]])]
    =============
    sage: Local_Information_Arbitrary_Ideal(K.ideal(3))
    [(Fractional ideal (3), [[a, 3/2*a + 1/2, -a, 1/2*a - 1/2],
    [1/2*a - 1/2, -3/2*a - 1/2, 1/2*a + 1/2, -1/2*a + 1/2],
    [a, 0, 1/2*a + 1/2, -a], [3/2*a + 1/2, 1/2*a - 1/2, 1/2*a + 1/2, -3/2*a - 1/2]])]
    =============
```

```
sage: Local_Information_Arbitrary_Ideal(K.ideal(6))
[(Fractional ideal (2), [[1/2*a + 1/2, a, 0, 3/2*a + 1/2],
[a, 1/2*a + 1/2, 0, a], [0, a, a, 0],
[3/2*a + 1/2, 3/2*a + 1/2, a, 3/2*a + 1/2]]),
(Fractional ideal (3), [[a, 3/2*a + 1/2, -a, 1/2*a - 1/2],
[1/2*a - 1/2, -3/2*a - 1/2, 1/2*a + 1/2, -1/2*a + 1/2],
[a, 0, 1/2*a + 1/2, -a], [3/2*a + 1/2, 1/2*a - 1/2, 1/2*a + 1/2, -3/2*a - 1/2]])]
=============


"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I != K.ideal(0)

POWERFACTORS = []
LOCALINFO=[]
for k in I.factor():
    POWERFACTORS.append(k[0]^(k[1]))
for k in POWERFACTORS:
```

```
            if k.norm()%2 == 0:
                LOCALINFO.append((k,Local_Isomorphism_Powers_of_Even_Prime(k)))
            if k.norm()%2 == 1:
                LOCALINFO.append((k,Local_Isomorphism_for_Powers_of_Odd_Prime_Ideal(k)))
        return LOCALINFO
```

```
    def Matrix_Embedding_with_CRT(I):
        """
        This will CRT the matrices found in the previous code.
        =============
        EXAMPLES:
        =============
        =============
        sage: Matrix_Embedding_with_CRT(K.ideal(2))
        [[1/2*a + 1/2, a, 0, 3/2*a + 1/2], [a, 1/2*a + 1/2, 0, a],
        [0, a, a, 0], [3/2*a + 1/2, 3/2*a + 1/2, a, 3/2*a + 1/2]]
        =============
```

```
sage: Matrix_Embedding_with_CRT(K.ideal(3))
[[a, 3/2*a + 1/2, -a, 1/2*a - 1/2],
[1/2*a - 1/2, -3/2*a - 1/2, 1/2*a + 1/2, -1/2*a + 1/2],
[a, 0, 1/2*a + 1/2, -a], [3/2*a + 1/2, 1/2*a - 1/2, 1/2*a + 1/2, -3/2*a - 1/2]]
==============
sage: Matrix_Embedding_with_CRT(K.ideal(6))
[[-1/2*a + 3/2, -1, 2*a, 1/2*a - 1/2],
[2*a + 1, 3/2*a - 1/2, -a - 1, -2*a - 1],
[-2*a, 3*a, 2*a - 1, 2*a], [3/2*a + 1/2, 1/2*a - 1/2, 2*a - 1, -3/2*a - 1/2]]
==============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)

FACTOR=I.factor()
INVERSES=[]
```

```
E1Matrix=[]
E2Matrix=[]
E3Matrix=[]
E4Matrix=[]



LocalInfo = Local_Information_Arbitrary_Ideal(I)


if len(LocalInfo) == 1:
    return LocalInfo[0][1]
else:
    for f in FACTOR:
        #This returns the inverse of M/m (i.e., b) mod m followed by M/m
        #(b,M/m)
        INVERSES.append((Inverse_mod_Ideal((I/(f[0]^(f[1]))).gens_reduced()[0],
        (f[0]^(f[1]))), (I/(f[0]^(f[1]))).gens_reduced()[0]))
    E1LocalInfo=[]
    E2LocalInfo=[]
    E3LocalInfo=[]
```

```
E4LocalInfo=[]
for i in range(len(LocalInfo)):
    #For each prime power, append Ei mod that prime power
    E1LocalInfo.append(LocalInfo[i][1][0])
    E2LocalInfo.append(LocalInfo[i][1][1])
    E3LocalInfo.append(LocalInfo[i][1][2])
    E4LocalInfo.append(LocalInfo[i][1][3])
#return E1LocalInfo, E2LocalInfo, E3LocalInfo, E4LocalInfo, INVERSES

for j in range(4):
    E1Matrix.append(Residue_mod_Ideal(sum(E1LocalInfo[i][j]*INVERSES[i][0]*INVERSES[i][1]
    for i in range(len(E1LocalInfo)))   , I))
    E2Matrix.append(Residue_mod_Ideal(sum(E2LocalInfo[i][j]*INVERSES[i][0]*INVERSES[i][1]
    for i in range(len(E2LocalInfo)))   , I))
    E3Matrix.append(Residue_mod_Ideal(sum(E3LocalInfo[i][j]*INVERSES[i][0]*INVERSES[i][1]
    for i in range(len(E3LocalInfo)))   , I))
    E4Matrix.append(Residue_mod_Ideal(sum(E4LocalInfo[i][j]*INVERSES[i][0]*INVERSES[i][1]
    for i in range(len(E4LocalInfo)))   , I))
    #E1Matrix.append(Residue_mod_Ideal(E1LocalInfo[i][0]*INVERSES[i][0]*INVERSES[i][1],I))
return [E1Matrix, E2Matrix, E3Matrix, E4Matrix]
```

```
def Set_of_Reductions_for_Even_Prime(b,I):
    """
    I is a power of 2. b is an integer.
    Returns all elements of I that are congruent to b mod 2.

    =============
    EXAMPLES:
    =============
    =============
    sage: Set_of_Reductions_for_Even_Prime(1, K.ideal(2))
    [a]
    =============
    =============
    sage: Set_of_Reductions_for_Even_Prime(1, K.ideal(8))
    [-4*a - 1, -2*a - 1, -1, 2*a - 1, -3*a, -a, a, 3*a,
    -2*a + 1, 1, 2*a + 1, 4*a + 1, -a + 2, a + 2, 3*a + 2, 5*a + 2]
    =============
```

```
=============
sage: Set_of_Reductions_for_Even_Prime(a, K.ideal(3))
Traceback (most recent call last):
...
AssertionError
=============
=============
sage: Set_of_Reductions_for_Even_Prime(1/2, K.ideal(2))
Traceback (most recent call last):
...
AssertionError
=============
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I != K.ideal(0)
assert len(I.factor()) == 1
assert I.norm()%2 == 0
```

```
    assert b.is_integral() == True


    Lifts = []
    Res = List_of_Residues(I)


    for i in Res:
        if Residue_mod_Ideal(i, K.ideal(2)) == Residue_mod_Ideal(b,K.ideal(2)):
            Lifts.append(i)
    return Lifts


##################################
## This section of code is meant##
## to build the necessary       ##
## commands for saving as       ##
## matrices the elements of     ##
## norm N.                      ##
##################################

def Matrix_Embedding_of_Units(I):
    """
```

```
We need to take the 120 units and write them as matrices
with coefficients in O_K/ I*O_K. This will be used
to create the fundamental domain.


=============
EXAMPLES:
=============
=============
sage: Matrix_Embedding_of_Units(K.ideal(0))
Traceback (most recent call last):
...
AssertionError
=============
sage: Matrix_Embedding_of_Units(K.ideal(3))
[(1/2*a - 1/2, -3/2*a - 1/2, a, a), (-3/2*a - 1/2, 0, 0, -3/2*a - 1/2),
(-1/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2),
(-1/2*a + 1/2, -1/2*a - 1/2, -a, a), (-3/2*a - 1/2, 3/2*a + 1/2, -3/2*a - 1/2, 0),
(3/2*a + 1/2, -a, 1/2*a + 1/2, -a), (-3/2*a - 1/2, -3/2*a - 1/2, 3/2*a + 1/2, 0),
(3/2*a + 1/2, 1/2*a + 1/2, -a, -a), (1/2*a - 1/2, a, -3/2*a - 1/2, a),
(1/2*a - 1/2, 1/2*a + 1/2, 1/2*a + 1/2, -3/2*a - 1/2),
(3/2*a + 1/2, 1/2*a - 1/2, 1/2*a - 1/2, 1/2*a + 1/2), (-1/2*a + 1/2, -a, -1/2*a - 1/2, a),
```

(-1/2*a - 1/2, 3/2*a + 1/2, a, -a), (1/2*a + 1/2, -3/2*a - 1/2, -1/2*a + 1/2, -1/2*a - 1/2),

(3/2*a + 1/2, 0, 3/2*a + 1/2, 3/2*a + 1/2), (-3/2*a - 1/2, a, a, 0),

(-1/2*a - 1/2, 1/2*a - 1/2, 1/2*a - 1/2, -3/2*a - 1/2),

(3/2*a + 1/2, 3/2*a + 1/2, 0, 3/2*a + 1/2),

(-3/2*a - 1/2, -1/2*a + 1/2, -1/2*a - 1/2, 3/2*a + 1/2), (1/2*a + 1/2, -a, -a, 0),

(3/2*a + 1/2, -3/2*a - 1/2, -3/2*a - 1/2, -3/2*a - 1/2),

(-3/2*a - 1/2, -1/2*a - 1/2, -1/2*a + 1/2, 3/2*a + 1/2), (-1/2*a - 1/2, a, 3/2*a + 1/2, -a),

(1/2*a + 1/2, -1/2*a + 1/2, -3/2*a - 1/2, -1/2*a - 1/2), (0, a, a, 1/2*a + 1/2),

(-a, -a, -3/2*a - 1/2, -1/2*a - 1/2), (a, a, 1/2*a + 1/2, -1/2*a + 1/2),

(3/2*a + 1/2, -3/2*a - 1/2, 0, 3/2*a + 1/2), (-a, 1/2*a - 1/2, 0, a),

(0, -a, -a, -3/2*a - 1/2), (a, 0, 3/2*a + 1/2, -a),

(-3/2*a - 1/2, -1/2*a - 1/2, -1/2*a - 1/2, 1/2*a - 1/2),

(3/2*a + 1/2, 0, -3/2*a - 1/2, 3/2*a + 1/2), (a, 3/2*a + 1/2, 0, -a),

(-a, -3/2*a - 1/2, -a, -1/2*a - 1/2), (a, -1/2*a + 1/2, -a, -1/2*a - 1/2),

(-a, -1/2*a - 1/2, a, 3/2*a + 1/2), (-3/2*a - 1/2, a, -1/2*a + 1/2, -a),

(3/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2, -3/2*a - 1/2), (-a, a, -1/2*a - 1/2, 3/2*a + 1/2),

(0, -a, -a, 1/2*a - 1/2), (a, 1/2*a + 1/2, a, -1/2*a + 1/2),

(-3/2*a - 1/2, -1/2*a + 1/2, a, -a), (-a, 0, 1/2*a - 1/2, a),

(a, -a, -1/2*a + 1/2, -1/2*a - 1/2), (1/2*a + 1/2, -1/2*a + 1/2, -a, -a),

(a, -a, 3/2*a + 1/2, 1/2*a - 1/2), (-a, 0, -1/2*a - 1/2, a),

(0, 3/2*a + 1/2, -3/2*a - 1/2, -3/2*a - 1/2),

(1/2*a - 1/2, 1/2*a + 1/2, -3/2*a - 1/2, -1/2*a + 1/2),

(a, a, -1/2*a + 1/2, 3/2*a + 1/2), (-1/2*a + 1/2, -a, -a, 0),

(0, -3/2*a - 1/2, 3/2*a + 1/2, -3/2*a - 1/2), (a, -1/2*a + 1/2, a, 3/2*a + 1/2),

(a, 1/2*a + 1/2, 0, -a), (-1/2*a - 1/2, a, 1/2*a - 1/2, a),

(0, -3/2*a - 1/2, 3/2*a + 1/2, 0), (-1/2*a + 1/2, 3/2*a + 1/2, -1/2*a - 1/2, 1/2*a - 1/2),

(1/2*a + 1/2, -3/2*a - 1/2, -3/2*a - 1/2, -1/2*a + 1/2),

(-a, -3/2*a - 1/2, a, -1/2*a + 1/2), (a, 3/2*a + 1/2, -a, 1/2*a - 1/2),

(-1/2*a - 1/2, 3/2*a + 1/2, 3/2*a + 1/2, 1/2*a - 1/2),

(1/2*a - 1/2, -3/2*a - 1/2, 1/2*a + 1/2, -1/2*a + 1/2), (0, 3/2*a + 1/2, -3/2*a - 1/2, 0),

(1/2*a + 1/2, -a, -1/2*a + 1/2, -a), (-a, -1/2*a - 1/2, 0, a),

(-a, 1/2*a - 1/2, -a, -3/2*a - 1/2), (0, 3/2*a + 1/2, -3/2*a - 1/2, 3/2*a + 1/2),

(1/2*a - 1/2, a, a, 0), (-a, -a, 1/2*a - 1/2, -3/2*a - 1/2),

(-1/2*a + 1/2, -1/2*a - 1/2, 3/2*a + 1/2, 1/2*a - 1/2),

(0, -3/2*a - 1/2, 3/2*a + 1/2, 3/2*a + 1/2), (a, 0, 1/2*a + 1/2, -a),

(-a, a, -3/2*a - 1/2, -1/2*a + 1/2), (-1/2*a - 1/2, 1/2*a - 1/2, a, a),

(-a, a, 1/2*a - 1/2, 1/2*a + 1/2), (a, 0, -1/2*a + 1/2, -a),

(3/2*a + 1/2, 1/2*a - 1/2, -a, a), (-a, -1/2*a - 1/2, -a, 1/2*a - 1/2),

(0, a, a, -1/2*a + 1/2), (a, -a, 1/2*a + 1/2, -3/2*a - 1/2),

(-3/2*a - 1/2, -3/2*a - 1/2, -3/2*a - 1/2, 3/2*a + 1/2), (3/2*a + 1/2, -a, 1/2*a - 1/2, a),

(a, 1/2*a + 1/2, -a, -3/2*a - 1/2), (-a, 1/2*a - 1/2, a, 1/2*a + 1/2),

(a, 3/2*a + 1/2, a, 1/2*a + 1/2), (-a, -3/2*a - 1/2, 0, a),

```
(-3/2*a - 1/2, 0, 3/2*a + 1/2, -3/2*a - 1/2),
(3/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2, -1/2*a + 1/2), (-a, 0, -3/2*a - 1/2, a),
(0, a, a, 3/2*a + 1/2), (a, -1/2*a + 1/2, 0, -a),
(-3/2*a - 1/2, 3/2*a + 1/2, 0, -3/2*a - 1/2), (-a, -a, -1/2*a - 1/2, 1/2*a - 1/2),
(a, a, 3/2*a + 1/2, 1/2*a + 1/2), (0, -a, -a, -1/2*a - 1/2),
(-1/2*a - 1/2, 1/2*a - 1/2, 3/2*a + 1/2, 1/2*a + 1/2), (1/2*a + 1/2, -a, -3/2*a - 1/2, a),
(3/2*a + 1/2, 1/2*a + 1/2, 1/2*a - 1/2, -3/2*a - 1/2),
(-3/2*a - 1/2, 3/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2),
(-1/2*a - 1/2, a, a, 0), (3/2*a + 1/2, 1/2*a - 1/2, 1/2*a + 1/2, -3/2*a - 1/2),
(-3/2*a - 1/2, -3/2*a - 1/2, 0, -3/2*a - 1/2),
(1/2*a + 1/2, -1/2*a + 1/2, -1/2*a + 1/2, 3/2*a + 1/2),
(3/2*a + 1/2, -a, -a, 0), (-3/2*a - 1/2, 0, -3/2*a - 1/2, -3/2*a - 1/2),
(-1/2*a - 1/2, 3/2*a + 1/2, 1/2*a - 1/2, 1/2*a + 1/2),
(1/2*a + 1/2, -3/2*a - 1/2, -a, a), (1/2*a - 1/2, a, 1/2*a + 1/2, -a),
(-3/2*a - 1/2, -1/2*a + 1/2, -1/2*a + 1/2, -1/2*a - 1/2),
(-1/2*a + 1/2, -1/2*a - 1/2, -1/2*a - 1/2, 3/2*a + 1/2), (-1/2*a + 1/2, -a, 3/2*a + 1/2, -a),
(-3/2*a - 1/2, -1/2*a - 1/2, a, a), (3/2*a + 1/2, 3/2*a + 1/2, -3/2*a - 1/2, 0),
(-3/2*a - 1/2, a, -1/2*a - 1/2, a), (3/2*a + 1/2, -3/2*a - 1/2, 3/2*a + 1/2, 0),
(1/2*a - 1/2, 1/2*a + 1/2, a, -a), (1/2*a - 1/2, -3/2*a - 1/2, -3/2*a - 1/2, -1/2*a - 1/2),
(3/2*a + 1/2, 0, 0, 3/2*a + 1/2), (-1/2*a + 1/2, 3/2*a + 1/2, -a, -a)]
=============
```

```
sage: Matrix_Embedding_of_Units(K.ideal(2))
[(3/2*a + 1/2, a, 0, 1/2*a + 1/2), (a, 0, 0, a), (1/2*a + 1/2, 3/2*a + 1/2, 0, 3/2*a + 1/2),
(3/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2, a), (1/2*a + 1/2, 0, 3/2*a + 1/2, 3/2*a + 1/2),
(0, 1/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2), (3/2*a + 1/2, 0, a, 1/2*a + 1/2),
(a, 1/2*a + 1/2, a, 3/2*a + 1/2), (0, a, a, a), (a, a, 1/2*a + 1/2, 3/2*a + 1/2),
(1/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2, a), (0, 3/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2),
(1/2*a + 1/2, 1/2*a + 1/2, 0, 3/2*a + 1/2), (a, a, 0, a), (3/2*a + 1/2, 0, 0, 1/2*a + 1/2),
(0, 3/2*a + 1/2, 1/2*a + 1/2, a), (a, 1/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2),
(1/2*a + 1/2, 0, 1/2*a + 1/2, 3/2*a + 1/2), (3/2*a + 1/2, 3/2*a + 1/2, a, 3/2*a + 1/2),
(0, a, a, 1/2*a + 1/2), (a, 0, a, a), (1/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2),
(0, 1/2*a + 1/2, 3/2*a + 1/2, a), (3/2*a + 1/2, a, 3/2*a + 1/2, 3/2*a + 1/2),
(1/2*a + 1/2, a, a, 0), (a, 1/2*a + 1/2, 3/2*a + 1/2, 0),
(1/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2, 0), (3/2*a + 1/2, 0, 1/2*a + 1/2, 1/2*a + 1/2),
(3/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2), (a, 3/2*a + 1/2, 1/2*a + 1/2, 0),
(0, 3/2*a + 1/2, 1/2*a + 1/2, 0), (3/2*a + 1/2, a, 1/2*a + 1/2, a),
(1/2*a + 1/2, 0, 0, 3/2*a + 1/2), (a, 3/2*a + 1/2, 0, a),
(3/2*a + 1/2, 1/2*a + 1/2, 0, 1/2*a + 1/2), (a, 3/2*a + 1/2, a, 1/2*a + 1/2),
(3/2*a + 1/2, 1/2*a + 1/2, a, a), (0, a, a, 3/2*a + 1/2), (a, 0, 3/2*a + 1/2, a),
(1/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2, 0), (3/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2, 0),
(a, 3/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2), (a, a, 3/2*a + 1/2, 1/2*a + 1/2),
(0, 1/2*a + 1/2, 3/2*a + 1/2, 0), (3/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2, 0),
```

(1/2*a + 1/2, 3/2*a + 1/2, a, a), (a, a, a, 0), (0, a, a, 0),

(1/2*a + 1/2, 0, a, 3/2*a + 1/2), (1/2*a + 1/2, 1/2*a + 1/2, a, 1/2*a + 1/2),

(3/2*a + 1/2, a, a, 0), (0, 1/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2),

(3/2*a + 1/2, 0, 3/2*a + 1/2, 1/2*a + 1/2), (1/2*a + 1/2, a, 3/2*a + 1/2, a),

(1/2*a + 1/2, a, 1/2*a + 1/2, 1/2*a + 1/2), (0, 3/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2),

(a, 0, 1/2*a + 1/2, a), (a, 1/2*a + 1/2, 0, a), (3/2*a + 1/2, 3/2*a + 1/2, 0, 1/2*a + 1/2),

(1/2*a + 1/2, a, 0, 3/2*a + 1/2), (1/2*a + 1/2, a, 0, 3/2*a + 1/2),

(3/2*a + 1/2, 3/2*a + 1/2, 0, 1/2*a + 1/2), (a, 1/2*a + 1/2, 0, a), (a, 0, 1/2*a + 1/2, a),

(0, 3/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2), (1/2*a + 1/2, a, 1/2*a + 1/2, 1/2*a + 1/2),

(1/2*a + 1/2, a, 3/2*a + 1/2, a), (3/2*a + 1/2, 0, 3/2*a + 1/2, 1/2*a + 1/2),

(0, 1/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2), (3/2*a + 1/2, a, a, 0),

(1/2*a + 1/2, 1/2*a + 1/2, a, 1/2*a + 1/2), (1/2*a + 1/2, 0, a, 3/2*a + 1/2), (0, a, a, 0),

(a, a, a, 0), (1/2*a + 1/2, 3/2*a + 1/2, a, a), (3/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2, 0),

(0, 1/2*a + 1/2, 3/2*a + 1/2, 0), (a, a, 3/2*a + 1/2, 1/2*a + 1/2),

(a, 3/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2), (3/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2, 0),

(1/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2, 0), (a, 0, 3/2*a + 1/2, a), (0, a, a, 3/2*a + 1/2),

(3/2*a + 1/2, 1/2*a + 1/2, a, a), (a, 3/2*a + 1/2, a, 1/2*a + 1/2),

(3/2*a + 1/2, 1/2*a + 1/2, 0, 1/2*a + 1/2), (a, 3/2*a + 1/2, 0, a),

(1/2*a + 1/2, 0, 0, 3/2*a + 1/2), (3/2*a + 1/2, a, 1/2*a + 1/2, a),

(0, 3/2*a + 1/2, 1/2*a + 1/2, 0), (a, 3/2*a + 1/2, 1/2*a + 1/2, 0),

(3/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2, 3/2*a + 1/2),

```
(3/2*a + 1/2, 0, 1/2*a + 1/2, 1/2*a + 1/2), (1/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2, 0),
(a, 1/2*a + 1/2, 3/2*a + 1/2, 0), (1/2*a + 1/2, a, a, 0),
(3/2*a + 1/2, a, 3/2*a + 1/2, 3/2*a + 1/2), (0, 1/2*a + 1/2, 3/2*a + 1/2, a),
(1/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2), (a, 0, a, a), (0, a, a, 1/2*a + 1/2),
(3/2*a + 1/2, 3/2*a + 1/2, a, 3/2*a + 1/2), (1/2*a + 1/2, 0, 1/2*a + 1/2, 3/2*a + 1/2),
(a, 1/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2), (0, 3/2*a + 1/2, 1/2*a + 1/2, a),
(3/2*a + 1/2, 0, 0, 1/2*a + 1/2), (a, a, 0, a), (1/2*a + 1/2, 1/2*a + 1/2, 0, 3/2*a + 1/2),
(0, 3/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2), (1/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2, a),
(a, a, 1/2*a + 1/2, 3/2*a + 1/2), (0, a, a, a), (a, 1/2*a + 1/2, a, 3/2*a + 1/2),
(3/2*a + 1/2, 0, a, 1/2*a + 1/2), (0, 1/2*a + 1/2, 3/2*a + 1/2, 1/2*a + 1/2),
(1/2*a + 1/2, 0, 3/2*a + 1/2, 3/2*a + 1/2), (3/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2, a),
(1/2*a + 1/2, 3/2*a + 1/2, 0, 3/2*a + 1/2), (a, 0, 0, a), (3/2*a + 1/2, a, 0, 1/2*a + 1/2)]
=============
sage: Matrix_Embedding_of_Units(K.ideal(5*a))
[(15/2*a + 11/2, -4*a - 2, 1/2*a - 3/2, -5/2*a - 9/2), (1, 0, 0, 1),
(2*a, 2*a + 3, 2*a + 3, 3*a + 1), (4*a + 3, -6*a - 4, -3/2*a - 7/2, -7/2*a - 5/2),
(-15/2*a - 11/2, -5/2*a - 1/2, -5/2*a - 3/2, -5*a - 6),
(3/2*a + 3/2, -3*a - 4, -3*a - 5, -a - 1),
(-15/2*a - 11/2, -5/2*a - 3/2, -5/2*a - 1/2, -5*a - 6),
(3/2*a + 3/2, -3*a - 5, -3*a - 4, -a - 1),
(15/2*a + 11/2, 1/2*a - 3/2, -4*a - 2, -5/2*a - 9/2),
```

(-3*a - 4, 5*a + 4, 5*a + 4, 7/2*a + 9/2),

(-6*a - 5, -11/2*a - 11/2, -11/2*a - 11/2, 13/2*a + 9/2),

(4*a + 3, -3/2*a - 7/2, -6*a - 4, -7/2*a - 5/2),

(3*a + 5, 3/2*a + 3/2, a + 1, -3*a - 4), (5*a + 4, 7/2*a + 9/2, 3*a + 4, -5*a - 4),

(-5/2*a - 1/2, -5*a - 6, 15/2*a + 11/2, 5/2*a + 3/2),

(11/2*a + 7/2, -1/2*a + 1/2, -1/2*a + 1/2, -1/2*a - 5/2),

(-13/2*a - 9/2, -11/2*a - 11/2, -11/2*a - 11/2, 6*a + 5),

(-5/2*a - 1/2, 15/2*a + 11/2, -5*a - 6, 5/2*a + 3/2), (-2*a - 3, 2*a, -3*a - 1, 2*a + 3),

(2*a + 1, -5/2*a - 3/2, -5/2*a - 3/2, -3/2*a - 1/2),

(15/2*a + 11/2, 5*a + 5, 5*a + 5, -15/2*a - 11/2), (-2*a - 3, -3*a - 1, 2*a, 2*a + 3),

(3*a + 5, a + 1, 3/2*a + 3/2, -3*a - 4), (5*a + 4, 3*a + 4, 7/2*a + 9/2, -5*a - 4),

(-3/2*a - 1/2, 5/2*a + 3/2, 5/2*a + 3/2, 2*a + 1), (-3*a - 4, -a - 1, -3/2*a - 3/2, 3*a + 5),

(-7/2*a - 5/2, 3/2*a + 7/2, 6*a + 4, 4*a + 3),

(5/2*a + 3/2, -15/2*a - 11/2, 5*a + 6, -5/2*a - 1/2),

(2*a + 2, -7/2*a - 5/2, -4*a - 2, -2*a - 2),

(-1/2*a - 5/2, 1/2*a - 1/2, 1/2*a - 1/2, 11/2*a + 7/2),

(-1/2*a + 1/2, -1/2*a - 5/2, -11/2*a - 7/2, 1/2*a - 1/2),

(7/2*a + 9/2, -5*a - 4, -5*a - 4, -3*a - 4),

(5/2*a + 3/2, 5*a + 6, -15/2*a - 11/2, -5/2*a - 1/2),

(-1/2*a + 1/2, -11/2*a - 7/2, -1/2*a - 5/2, 1/2*a - 1/2),

(-3*a - 4, -3/2*a - 3/2, -a - 1, 3*a + 5), (3/2*a + 7/2, 4*a + 3, 7/2*a + 5/2, -6*a - 4),

(-a - 1, 3*a + 5, 3*a + 4, 3/2*a + 3/2),

(-4*a - 2, -5/2*a - 9/2, -15/2*a - 11/2, -1/2*a + 3/2),

(-5*a - 5, 15/2*a + 11/2, 15/2*a + 11/2, 5*a + 5), (-a - 1, 3*a + 4, 3*a + 5, 3/2*a + 3/2),

(4*a + 2, 2*a + 2, 2*a + 2, -7/2*a - 5/2), (-7/2*a - 5/2, 6*a + 4, 3/2*a + 7/2, 4*a + 3),

(-4*a - 2, -15/2*a - 11/2, -5/2*a - 9/2, -1/2*a + 3/2),

(2*a + 2, -4*a - 2, -7/2*a - 5/2, -2*a - 2), (3/2*a + 7/2, 7/2*a + 5/2, 4*a + 3, -6*a - 4),

(6*a + 4, 4*a + 3, 7/2*a + 5/2, -3/2*a - 7/2),

(-5/2*a - 9/2, -1/2*a + 3/2, 4*a + 2, 15/2*a + 11/2),

(-5/2*a - 3/2, -3/2*a - 1/2, -2*a - 1, 5/2*a + 3/2),

(-5*a - 6, 5/2*a + 3/2, 5/2*a + 1/2, -15/2*a - 11/2),

(-11/2*a - 11/2, 13/2*a + 9/2, 6*a + 5, 11/2*a + 11/2),

(1/2*a - 3/2, -5/2*a - 9/2, -15/2*a - 11/2, 4*a + 2),

(7/2*a + 5/2, 2*a + 2, 2*a + 2, -4*a - 2),

(-5*a - 6, 5/2*a + 1/2, 5/2*a + 3/2, -15/2*a - 11/2),

(1/2*a - 3/2, -15/2*a - 11/2, -5/2*a - 9/2, 4*a + 2),

(5/2*a + 3/2, 2*a + 1, 3/2*a + 1/2, -5/2*a - 3/2),

(-6*a - 4, -7/2*a - 5/2, -4*a - 3, 3/2*a + 7/2), (0, 1, -1, 0),

(11/2*a + 11/2, -6*a - 5, -13/2*a - 9/2, -11/2*a - 11/2),

(3*a + 1, -2*a - 3, -2*a - 3, 2*a), (5/2*a + 9/2, -4*a - 2, 1/2*a - 3/2, -15/2*a - 11/2),

(-5/2*a - 9/2, 4*a + 2, -1/2*a + 3/2, 15/2*a + 11/2), (-3*a - 1, 2*a + 3, 2*a + 3, -2*a),

(-11/2*a - 11/2, 6*a + 5, 13/2*a + 9/2, 11/2*a + 11/2), (0, -1, 1, 0),

```
(6*a + 4, 7/2*a + 5/2, 4*a + 3, -3/2*a - 7/2),
(-5/2*a - 3/2, -2*a - 1, -3/2*a - 1/2, 5/2*a + 3/2),
(-1/2*a + 3/2, 15/2*a + 11/2, 5/2*a + 9/2, -4*a - 2),
(5*a + 6, -5/2*a - 1/2, -5/2*a - 3/2, 15/2*a + 11/2),
(-7/2*a - 5/2, -2*a - 2, -2*a - 2, 4*a + 2),
(-1/2*a + 3/2, 5/2*a + 9/2, 15/2*a + 11/2, -4*a - 2),
(11/2*a + 11/2, -13/2*a - 9/2, -6*a - 5, -11/2*a - 11/2),
(5*a + 6, -5/2*a - 3/2, -5/2*a - 1/2, 15/2*a + 11/2),
(5/2*a + 3/2, 3/2*a + 1/2, 2*a + 1, -5/2*a - 3/2),
(5/2*a + 9/2, 1/2*a - 3/2, -4*a - 2, -15/2*a - 11/2),
(-6*a - 4, -4*a - 3, -7/2*a - 5/2, 3/2*a + 7/2),
(-3/2*a - 7/2, -7/2*a - 5/2, -4*a - 3, 6*a + 4), (-2*a - 2, 4*a + 2, 7/2*a + 5/2, 2*a + 2),
(4*a + 2, 15/2*a + 11/2, 5/2*a + 9/2, 1/2*a - 3/2),
(7/2*a + 5/2, -6*a - 4, -3/2*a - 7/2, -4*a - 3),
(-4*a - 2, -2*a - 2, -2*a - 2, 7/2*a + 5/2), (a + 1, -3*a - 4, -3*a - 5, -3/2*a - 3/2),
(5*a + 5, -15/2*a - 11/2, -15/2*a - 11/2, -5*a - 5),
(4*a + 2, 5/2*a + 9/2, 15/2*a + 11/2, 1/2*a - 3/2),
(a + 1, -3*a - 5, -3*a - 4, -3/2*a - 3/2), (-3/2*a - 7/2, -4*a - 3, -7/2*a - 5/2, 6*a + 4),
(3*a + 4, 3/2*a + 3/2, a + 1, -3*a - 5),
(1/2*a - 1/2, 11/2*a + 7/2, 1/2*a + 5/2, -1/2*a + 1/2),
(-5/2*a - 3/2, -5*a - 6, 15/2*a + 11/2, 5/2*a + 1/2),
```

-7/2*a - 9/2, 5*a + 4, 5*a + 4, 3*a + 4),

(1/2*a - 1/2, 1/2*a + 5/2, 11/2*a + 7/2, -1/2*a + 1/2),

(1/2*a + 5/2, -1/2*a + 1/2, -1/2*a + 1/2, -11/2*a - 7/2),

(-2*a - 2, 7/2*a + 5/2, 4*a + 2, 2*a + 2),

(-5/2*a - 3/2, 15/2*a + 11/2, -5*a - 6, 5/2*a + 1/2),

(7/2*a + 5/2, -3/2*a - 7/2, -6*a - 4, -4*a - 3), (3*a + 4, a + 1, 3/2*a + 3/2, -3*a - 5),

(3/2*a + 1/2, -5/2*a - 3/2, -5/2*a - 3/2, -2*a - 1),

(-5*a - 4, -3*a - 4, -7/2*a - 9/2, 5*a + 4), (-3*a - 5, -a - 1, -3/2*a - 3/2, 3*a + 4),

(2*a + 3, 3*a + 1, -2*a, -2*a - 3), (-15/2*a - 11/2, -5*a - 5, -5*a - 5, 15/2*a + 11/2),

(-2*a - 1, 5/2*a + 3/2, 5/2*a + 3/2, 3/2*a + 1/2), (2*a + 3, -2*a, 3*a + 1, -2*a - 3),

(5/2*a + 1/2, -15/2*a - 11/2, 5*a + 6, -5/2*a - 3/2),

(13/2*a + 9/2, 11/2*a + 11/2, 11/2*a + 11/2, -6*a - 5),

(-11/2*a - 7/2, 1/2*a - 1/2, 1/2*a - 1/2, 1/2*a + 5/2),

(5/2*a + 1/2, 5*a + 6, -15/2*a - 11/2, -5/2*a - 3/2),

(-5*a - 4, -7/2*a - 9/2, -3*a - 4, 5*a + 4), (-3*a - 5, -3/2*a - 3/2, -a - 1, 3*a + 4),

(-4*a - 3, 3/2*a + 7/2, 6*a + 4, 7/2*a + 5/2),

(6*a + 5, 11/2*a + 11/2, 11/2*a + 11/2, -13/2*a - 9/2),

(3*a + 4, -5*a - 4, -5*a - 4, -7/2*a - 9/2),

(-15/2*a - 11/2, -1/2*a + 3/2, 4*a + 2, 5/2*a + 9/2),

(-3/2*a - 3/2, 3*a + 5, 3*a + 4, a + 1), (15/2*a + 11/2, 5/2*a + 3/2, 5/2*a + 1/2, 5*a + 6),

(-3/2*a - 3/2, 3*a + 4, 3*a + 5, a + 1), (15/2*a + 11/2, 5/2*a + 1/2, 5/2*a + 3/2, 5*a + 6),

```
(-4*a - 3, 6*a + 4, 3/2*a + 7/2, 7/2*a + 5/2), (-2*a, -2*a - 3, -2*a - 3, -3*a - 1),
(-1, 0, 0, -1), (-15/2*a - 11/2, 4*a + 2, -1/2*a + 3/2, 5/2*a + 9/2)]
==============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)

BASIS = Matrix_Embedding_with_CRT(I)
U = Representations_in_Maximal_Order_Quadruple(1)

E1SET=[]
E2SET=[]
E3SET=[]
E4SET=[]
EIEIO=[]

for i in U:
```

```
        E1SET.append([Residue_mod_Ideal(BASIS[0][0]*i[0],I),
        Residue_mod_Ideal(BASIS[0][1]*i[0],I),
        Residue_mod_Ideal(BASIS[0][2]*i[0],I),
        Residue_mod_Ideal(BASIS[0][3]*i[0],I)])
        E2SET.append([Residue_mod_Ideal(BASIS[1][0]*i[1],I),
        Residue_mod_Ideal(BASIS[1][1]*i[1],I),
        Residue_mod_Ideal(BASIS[1][2]*i[1],I),
        Residue_mod_Ideal(BASIS[1][3]*i[1],I)])
        E3SET.append([Residue_mod_Ideal(BASIS[2][0]*i[2],I),
        Residue_mod_Ideal(BASIS[2][1]*i[2],I),
        Residue_mod_Ideal(BASIS[2][2]*i[2],I),
        Residue_mod_Ideal(BASIS[2][3]*i[2],I)])
        E4SET.append([Residue_mod_Ideal(BASIS[3][0]*i[3],I),
        Residue_mod_Ideal(BASIS[3][1]*i[3],I),
        Residue_mod_Ideal(BASIS[3][2]*i[3],I),
        Residue_mod_Ideal(BASIS[3][3]*i[3],I)])
    for j in range(len(E1SET)):
        EIEIO.append((Residue_mod_Ideal(E1SET[j][0] + E2SET[j][0] + E3SET[j][0] + E4SET[j][0], I),
        Residue_mod_Ideal(E1SET[j][1] + E2SET[j][1] + E3SET[j][1] + E4SET[j][1],I),
        Residue_mod_Ideal(E1SET[j][2] + E2SET[j][2] + E3SET[j][2] + E4SET[j][2],I),
        Residue_mod_Ideal(E1SET[j][3] + E2SET[j][3] + E3SET[j][3] + E4SET[j][3],I)))
```

```
    return EIEIO




def Matrix_Embedding_of_Units_Theta_p(p,I):
    """
    We need to take the elements in Theta_p
    and write them as matrices with coefficients
    in O_K/I*O_K.

    ==============
    EXAMPLES:
    ==============
    sage: Matrix_Embedding_of_Units_Theta_p(2, K.ideal(3))
    [[-a, -3/2*a - 1/2, -1/2*a + 1/2, -1/2*a + 1/2], [a, 1/2*a - 1/2, 0, a],
    [3/2*a + 1/2, -1/2*a + 1/2, -1/2*a - 1/2, 0],
    [-a, 3/2*a + 1/2, -1/2*a - 1/2, -1/2*a - 1/2],
```

```
[-3/2*a - 1/2, 1/2*a + 1/2, 1/2*a + 1/2, 1/2*a + 1/2]]
==============
sage: Matrix_Embedding_of_Units_Theta_p(3, K.ideal(2))
[[1/2*a + 1/2, a, 0, 3/2*a + 1/2], [a, 1/2*a + 1/2, 0, a],
[3/2*a + 1/2, 3/2*a + 1/2, 0, 1/2*a + 1/2], [0, 1/2*a + 1/2, 3/2*a + 1/2, 3/2*a + 1/2],
[3/2*a + 1/2, 0, 3/2*a + 1/2, 1/2*a + 1/2], [1/2*a + 1/2, a, 3/2*a + 1/2, a],
[0, a, a, 0], [a, a, a, 0], [3/2*a + 1/2, a, a, 0], [a, 0, 1/2*a + 1/2, a]]
==============
sage: Matrix_Embedding_of_Units_Theta_p(2, K.ideal(a))
[[0, -1/2*a - 1/2, -a - 1, -1/2*a - 1/2], [-a - 1, -a - 1, -1/2*a - 1/2, 0],
[1/2*a + 1/2, 0, -a - 1, -a - 1], [a + 1, 1/2*a + 1/2, a + 1, 0],
[a + 1, -a - 1, -a - 1, 1/2*a + 1/2]]
==============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)
```

```
TPMAT=[]



REP = Representations_in_Max_Order_Trad_Basis(p)

P=Representatives_up_to_Unit(REP)

THETAP= Representatives_up_to_Unit_e_Basis(P)

EIBASIS = Matrix_Embedding_with_CRT(I)




TESTE1=[]

TESTE2=[]

TESTE3=[]

TESTE4=[]

SUM=[]

SUMMOD=[]


for i in THETAP:

    TESTE1.append([EIBASIS[0][0]*i[0][0], EIBASIS[0][1]*i[0][0],

    EIBASIS[0][2]*i[0][0], EIBASIS[0][3]*i[0][0]])

    TESTE2.append([EIBASIS[1][0]*i[0][1], EIBASIS[1][1]*i[0][1],
```

```
    EIBASIS[1][2]*i[0][1], EIBASIS[1][3]*i[0][1]])
    TESTE3.append([EIBASIS[2][0]*i[0][2], EIBASIS[2][1]*i[0][2],
    EIBASIS[2][2]*i[0][2], EIBASIS[2][3]*i[0][2]])
    TESTE4.append([EIBASIS[3][0]*i[0][3], EIBASIS[3][1]*i[0][3],
    EIBASIS[3][2]*i[0][3], EIBASIS[3][3]*i[0][3]])


for i in range(len(TESTE1)):
    SUM.append([TESTE1[i][0]+TESTE2[i][0]+TESTE3[i][0]+TESTE4[i][0],
    TESTE1[i][1]+TESTE2[i][1]+TESTE3[i][1]+TESTE4[i][1],
    TESTE1[i][2]+TESTE2[i][2]+TESTE3[i][2]+TESTE4[i][2],
    TESTE1[i][3]+TESTE2[i][3]+TESTE3[i][3]+TESTE4[i][3] ])

for i in SUM:
    SUMMOD.append([Residue_mod_Ideal(i[0], I),
    Residue_mod_Ideal(i[1],I), Residue_mod_Ideal(i[2],I),
    Residue_mod_Ideal(i[3],I)])

return SUMMOD
```

```
###################################
## This section of code is meant##
## to build the necessary       ##
## commands for computing       ##
## the fundamental domain.      ##
## CHI = 1.                     ##
###################################

def Singular_Orbit((A,B),I):
    """
    Looks at the orbit of a single point in P^1(O_K/I*O_K) under the action of the units.

    THIS ACTION OF UNITS WILL BE IN O_K/I*O_K TOO.
    =============
    =============
    EXAMPLES:
    =============
    sage: Singular_Orbit((1,0), K.ideal(5+2*(1+a)/2))
    [(12*a, 5*a), (1, 0), (8*a, 5*a), (7*a, 5*a), (-11*a, 5*a), (-10*a,
    5*a), (0, 5*a), (-13*a, 5*a), (-9*a, 5*a), (-8*a, 5*a), (-7*a, 5*a),
```

```
(-15*a, 5*a)]
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

#NEED TO ASSERT SOMEHOW THAT (a,b) IS AN ELEMENT OF P^1(O_K/I*O_K).

U = Matrix_Embedding_of_Units(I)

ORBIT=[]
for u in U:
    if ORBIT.count(Projective_Space_Representative((u[0]*A+u[1]*B, u[2]*A + u[3]*B),I)) == 0:
        ORBIT.append(Projective_Space_Representative((u[0]*A+u[1]*B, u[2]*A + u[3]*B),I))

return ORBIT
```

```
def Fundamental_Domain_Orbits(I):
    """
    looks at P^1(O_K/I*O_K). considers the action of the units on that set of points.
    returns the number of orbits, and the orbits themselves.
    THIS IS FOR TRIVIAL CHARACTER!

    Note with the examples: it is not until an ideal of norm 29 that there
    are any cusp forms with trivial character. Thus, there should be only
    one orbit for any ideal of norm <29. Moreover, for the ideal (5+2*(1+a)/2),
    Dembele shows there is only one dimension to the cusp subspace.
    =============
    EXAMPLES:
    =============
    sage: Fundamental_Domain_Orbits(K.ideal(2))
    [[(1, 0), (a, a), (3/2*a + 1/2, a), (0, a), (1/2*a + 1/2, a)]]
    =============
    sage: Fundamental_Domain_Orbits(K.ideal(4))
    [[(1, 3*a + 1), (1, 0), (1, 1), (-1/2*a + 1/2, 1), (3*a + 1, 1),
    (3/2*a + 1/2, 1), (2*a + 1, 1), (a + 1, 1), (-a, 1), (1, 2*a),
    (-3/2*a - 1/2, 1), (0, 1), (3/2*a - 1/2, 1), (-1/2*a - 1/2, 1),
```

```
(1/2*a + 1/2, 1), (2*a, 1), (1, a + 1), (1/2*a - 1/2, 1),
(5/2*a + 1/2, 1), (a, 1)]]
==============
sage: Fundamental_Domain_Orbits(K.ideal(5+2*(1+a)/2))
[[(12*a, 5*a), (1, 0), (8*a, 5*a), (7*a, 5*a), (-11*a, 5*a), (-10*a,
5*a), (0, 5*a), (-13*a, 5*a), (-9*a, 5*a), (-8*a, 5*a), (-7*a, 5*a),
(-15*a, 5*a)], [(-6*a, 5*a), (-14*a, 5*a), (6*a, 5*a), (15*a, 5*a),
(4*a, 5*a), (-4*a, 5*a), (a, 5*a), (5*a, 5*a), (14*a, 5*a), (-2*a, 5*a),
(3*a, 5*a), (9*a, 5*a), (-12*a, 5*a), (13*a, 5*a), (11*a, 5*a), (10*a,
5*a), (-a, 5*a), (-3*a, 5*a), (-5*a, 5*a), (2*a, 5*a)]]
==============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)

PONE = Projective_Space_Arbitrary(I)
D= Projective_Space_Arbitrary(I)
```

```
    Set=[]



    while len(D) >0:
        Set.append(Singular_Orbit(D[0],I))
        for i in Singular_Orbit(D[0],I):
                D.remove(i)
    return  Set
```



```
###################################
## This section of code is meant##
## to build the necessary       ##
## commands for computing       ##
## the fundamental domain.      ##
## CHI neq 1.                   ##
###################################
```

```
def CHI(I):
    """
    For p lying above P
    chi(p)= 1 if P equiv 1,3,7,9,13,17,23,27,31,33,37,39 (40)
    chi(p)=-1 if P equiv 5,11,19,21,29
    chi(p) =0 if P =2.

    This is equivalent, actually, to:
    chi(p) =1 if N(p) equiv 1, -1 (8)
    chi(p)=-1 if N(p) equiv 3,-3 (8)
    chi(p)=0 otherwise.

    =============
    EXAMPLES:
    =============
    sage: CHI(K.ideal(2))
    0
    =============
    sage: CHI(K.ideal(3))
    1
```

```
=============
sage: CHI(K.ideal(a))
-1
=============
sage: CHI(K.ideal(5))
1
=============
sage: CHI(K.ideal(123*a))
-1
=============
"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)
Z = I.factor()



M=[]
```

```
    for i in Z:
        if i[0].norm() %2 == 0 %2:
            M.append(0)
        if i[0].norm() %8 == 3 %8:
            M.append((-1)^(i[1]))
        if i[0].norm() %8 == -3 %8:
            M.append((-1)^(i[1]))
        if i[0].norm() %8 == 1 %8 >0:
            M.append(1)
        if i[0].norm() %8 == -1 %8:
            M.append(1)
    return prod(M)



def Hone(I):
    """
    In Dembele Hone(I):= {(a,b) in (O_K/I)^2
    with gcd (a,b) invertible in O_K/I}.
```

```
N.B.: on ANY prime ideal, this should return
a set only excluding (0,0). So, the size should be N(p)^2-1.

=============
EXAMPLES:
=============
sage: Hone(K.ideal(2))
[(0, a), (0, 1/2*a + 1/2), (0, 3/2*a + 1/2), (a, 0), (a, a), (a, 1/2*a + 1/2),
(a, 3/2*a + 1/2), (1/2*a + 1/2, 0), (1/2*a + 1/2, a), (1/2*a + 1/2, 1/2*a + 1/2),
(1/2*a + 1/2, 3/2*a + 1/2), (3/2*a + 1/2, 0), (3/2*a + 1/2, a), (3/2*a + 1/2, 1/2*a + 1/2),
(3/2*a + 1/2, 3/2*a + 1/2)]
=============
=============
sage: Hone(K.ideal(3))
[(-3/2*a - 1/2, -3/2*a - 1/2), (-3/2*a - 1/2, -1/2*a - 1/2), (-3/2*a - 1/2, 1/2*a - 1/2),
(-3/2*a - 1/2, -a), (-3/2*a - 1/2, 0), (-3/2*a - 1/2, a), (-3/2*a - 1/2, -1/2*a + 1/2),
(-3/2*a - 1/2, 1/2*a + 1/2), (-3/2*a - 1/2, 3/2*a + 1/2), (-1/2*a - 1/2, -3/2*a - 1/2),
(-1/2*a - 1/2, -1/2*a - 1/2), (-1/2*a - 1/2, 1/2*a - 1/2), (-1/2*a - 1/2, -a), (-1/2*a - 1/2, 0),
(-1/2*a - 1/2, a), (-1/2*a - 1/2, -1/2*a + 1/2), (-1/2*a - 1/2, 1/2*a + 1/2),
(-1/2*a - 1/2, 3/2*a + 1/2), (1/2*a - 1/2, -3/2*a - 1/2), (1/2*a - 1/2, -1/2*a - 1/2),
(1/2*a - 1/2, 1/2*a - 1/2), (1/2*a - 1/2, -a), (1/2*a - 1/2, 0), (1/2*a - 1/2, a),
```

```
(1/2*a - 1/2, -1/2*a + 1/2), (1/2*a - 1/2, 1/2*a + 1/2), (1/2*a - 1/2, 3/2*a + 1/2),
(-a, -3/2*a - 1/2), (-a, -1/2*a - 1/2), (-a, 1/2*a - 1/2), (-a, -a), (-a, 0), (-a, a),
(-a, -1/2*a + 1/2), (-a, 1/2*a + 1/2), (-a, 3/2*a + 1/2), (0, -3/2*a - 1/2), (0, -1/2*a - 1/2),
(0, 1/2*a - 1/2), (0, -a), (0, a), (0, -1/2*a + 1/2), (0, 1/2*a + 1/2), (0, 3/2*a + 1/2),
(a, -3/2*a - 1/2), (a, -1/2*a - 1/2), (a, 1/2*a - 1/2), (a, -a), (a, 0), (a, a), (a, -1/2*a + 1/2),
(a, 1/2*a + 1/2), (a, 3/2*a + 1/2), (-1/2*a + 1/2, -3/2*a - 1/2), (-1/2*a + 1/2, -1/2*a - 1/2),
(-1/2*a + 1/2, 1/2*a - 1/2), (-1/2*a + 1/2, -a), (-1/2*a + 1/2, 0), (-1/2*a + 1/2, a),
(-1/2*a + 1/2, -1/2*a + 1/2), (-1/2*a + 1/2, 1/2*a + 1/2), (-1/2*a + 1/2, 3/2*a + 1/2),
(1/2*a + 1/2, -3/2*a - 1/2), (1/2*a + 1/2, -1/2*a - 1/2), (1/2*a + 1/2, 1/2*a - 1/2),
(1/2*a + 1/2, -a), (1/2*a + 1/2, 0), (1/2*a + 1/2, a), (1/2*a + 1/2, -1/2*a + 1/2),
(1/2*a + 1/2, 1/2*a + 1/2), (1/2*a + 1/2, 3/2*a + 1/2), (3/2*a + 1/2, -3/2*a - 1/2),
(3/2*a + 1/2, -1/2*a - 1/2), (3/2*a + 1/2, 1/2*a - 1/2), (3/2*a + 1/2, -a), (3/2*a + 1/2, 0),
(3/2*a + 1/2, a), (3/2*a + 1/2, -1/2*a + 1/2), (3/2*a + 1/2, 1/2*a + 1/2), (3/2*a + 1/2, 3/2*a + 1/2)]
==============
==============
sage: Hone(K.ideal(4))
[(-3/2*a - 1/2, -3/2*a - 1/2), (-3/2*a - 1/2, -1/2*a - 1/2), (-3/2*a - 1/2, 1/2*a - 1/2),
(-3/2*a - 1/2, 3/2*a - 1/2), (-3/2*a - 1/2, -a), (-3/2*a - 1/2, 0), (-3/2*a - 1/2, a),
(-3/2*a - 1/2, 2*a), (-3/2*a - 1/2, -1/2*a + 1/2), (-3/2*a - 1/2, 1/2*a + 1/2),
(-3/2*a - 1/2, 3/2*a + 1/2), (-3/2*a - 1/2, 5/2*a + 1/2), (-3/2*a - 1/2, 1),
(-3/2*a - 1/2, a + 1), (-3/2*a - 1/2, 2*a + 1), (-3/2*a - 1/2, 3*a + 1),
```

(-1/2*a - 1/2, -3/2*a - 1/2), (-1/2*a - 1/2, -1/2*a - 1/2), (-1/2*a - 1/2, 1/2*a - 1/2),
(-1/2*a - 1/2, 3/2*a - 1/2), (-1/2*a - 1/2, -a), (-1/2*a - 1/2, 0), (-1/2*a - 1/2, a),
(-1/2*a - 1/2, 2*a), (-1/2*a - 1/2, -1/2*a + 1/2), (-1/2*a - 1/2, 1/2*a + 1/2),
(-1/2*a - 1/2, 3/2*a + 1/2), (-1/2*a - 1/2, 5/2*a + 1/2), (-1/2*a - 1/2, 1),
(-1/2*a - 1/2, a + 1), (-1/2*a - 1/2, 2*a + 1), (-1/2*a - 1/2, 3*a + 1),
(1/2*a - 1/2, -3/2*a - 1/2), (1/2*a - 1/2, -1/2*a - 1/2), (1/2*a - 1/2, 1/2*a - 1/2),
(1/2*a - 1/2, 3/2*a - 1/2), (1/2*a - 1/2, -a), (1/2*a - 1/2, 0), (1/2*a - 1/2, a),
(1/2*a - 1/2, 2*a), (1/2*a - 1/2, -1/2*a + 1/2), (1/2*a - 1/2, 1/2*a + 1/2),
(1/2*a - 1/2, 3/2*a + 1/2), (1/2*a - 1/2, 5/2*a + 1/2), (1/2*a - 1/2, 1),
(1/2*a - 1/2, a + 1), (1/2*a - 1/2, 2*a + 1), (1/2*a - 1/2, 3*a + 1),
(3/2*a - 1/2, -3/2*a - 1/2), (3/2*a - 1/2, -1/2*a - 1/2), (3/2*a - 1/2, 1/2*a - 1/2),
(3/2*a - 1/2, 3/2*a - 1/2), (3/2*a - 1/2, -a), (3/2*a - 1/2, 0), (3/2*a - 1/2, a),
(3/2*a - 1/2, 2*a), (3/2*a - 1/2, -1/2*a + 1/2), (3/2*a - 1/2, 1/2*a + 1/2),
(3/2*a - 1/2, 3/2*a + 1/2), (3/2*a - 1/2, 5/2*a + 1/2), (3/2*a - 1/2, 1),
(3/2*a - 1/2, a + 1), (3/2*a - 1/2, 2*a + 1), (3/2*a - 1/2, 3*a + 1),
(-a, -3/2*a - 1/2), (-a, -1/2*a - 1/2), (-a, 1/2*a - 1/2), (-a, 3/2*a - 1/2), (-a, -a),
(-a, 0), (-a, a), (-a, 2*a), (-a, -1/2*a + 1/2), (-a, 1/2*a + 1/2), (-a, 3/2*a + 1/2),
(-a, 5/2*a + 1/2), (-a, 1), (-a, a + 1), (-a, 2*a + 1), (-a, 3*a + 1), (0, -3/2*a - 1/2),
(0, -1/2*a - 1/2), (0, 1/2*a - 1/2), (0, 3/2*a - 1/2), (0, -a), (0, a), (0, -1/2*a + 1/2),
(0, 1/2*a + 1/2), (0, 3/2*a + 1/2), (0, 5/2*a + 1/2), (0, 1), (0, 2*a + 1),
(a, -3/2*a - 1/2), (a, -1/2*a - 1/2), (a, 1/2*a - 1/2), (a, 3/2*a - 1/2), (a, -a),

(a, 0), (a, a), (a, 2*a), (a, -1/2*a + 1/2), (a, 1/2*a + 1/2), (a, 3/2*a + 1/2),
(a, 5/2*a + 1/2), (a, 1), (a, a + 1), (a, 2*a + 1), (a, 3*a + 1), (2*a, -3/2*a - 1/2),
(2*a, -1/2*a - 1/2), (2*a, 1/2*a - 1/2), (2*a, 3/2*a - 1/2), (2*a, -a), (2*a, a),
(2*a, -1/2*a + 1/2), (2*a, 1/2*a + 1/2), (2*a, 3/2*a + 1/2), (2*a, 5/2*a + 1/2),
(2*a, 1), (2*a, 2*a + 1), (-1/2*a + 1/2, -3/2*a - 1/2), (-1/2*a + 1/2, -1/2*a - 1/2),
(-1/2*a + 1/2, 1/2*a - 1/2), (-1/2*a + 1/2, 3/2*a - 1/2), (-1/2*a + 1/2, -a),
(-1/2*a + 1/2, 0), (-1/2*a + 1/2, a), (-1/2*a + 1/2, 2*a), (-1/2*a + 1/2, -1/2*a + 1/2),
(-1/2*a + 1/2, 1/2*a + 1/2), (-1/2*a + 1/2, 3/2*a + 1/2), (-1/2*a + 1/2, 5/2*a + 1/2),
(-1/2*a + 1/2, 1), (-1/2*a + 1/2, a + 1), (-1/2*a + 1/2, 2*a + 1),
(-1/2*a + 1/2, 3*a + 1), (1/2*a + 1/2, -3/2*a - 1/2), (1/2*a + 1/2, -1/2*a - 1/2),
(1/2*a + 1/2, 1/2*a - 1/2), (1/2*a + 1/2, 3/2*a - 1/2), (1/2*a + 1/2, -a),
(1/2*a + 1/2, 0), (1/2*a + 1/2, a), (1/2*a + 1/2, 2*a), (1/2*a + 1/2, -1/2*a + 1/2),
(1/2*a + 1/2, 1/2*a + 1/2), (1/2*a + 1/2, 3/2*a + 1/2), (1/2*a + 1/2, 5/2*a + 1/2),
(1/2*a + 1/2, 1), (1/2*a + 1/2, a + 1), (1/2*a + 1/2, 2*a + 1), (1/2*a + 1/2, 3*a + 1),
(3/2*a + 1/2, -3/2*a - 1/2), (3/2*a + 1/2, -1/2*a - 1/2), (3/2*a + 1/2, 1/2*a - 1/2),
(3/2*a + 1/2, 3/2*a - 1/2), (3/2*a + 1/2, -a), (3/2*a + 1/2, 0), (3/2*a + 1/2, a),
(3/2*a + 1/2, 2*a), (3/2*a + 1/2, -1/2*a + 1/2), (3/2*a + 1/2, 1/2*a + 1/2),
(3/2*a + 1/2, 3/2*a + 1/2), (3/2*a + 1/2, 5/2*a + 1/2), (3/2*a + 1/2, 1),
(3/2*a + 1/2, a + 1), (3/2*a + 1/2, 2*a + 1), (3/2*a + 1/2, 3*a + 1),
(5/2*a + 1/2, -3/2*a - 1/2), (5/2*a + 1/2, -1/2*a - 1/2), (5/2*a + 1/2, 1/2*a - 1/2),
(5/2*a + 1/2, 3/2*a - 1/2), (5/2*a + 1/2, -a), (5/2*a + 1/2, 0), (5/2*a + 1/2, a),

```
(5/2*a + 1/2, 2*a), (5/2*a + 1/2, -1/2*a + 1/2), (5/2*a + 1/2, 1/2*a + 1/2),
(5/2*a + 1/2, 3/2*a + 1/2), (5/2*a + 1/2, 5/2*a + 1/2), (5/2*a + 1/2, 1),
(5/2*a + 1/2, a + 1), (5/2*a + 1/2, 2*a + 1), (5/2*a + 1/2, 3*a + 1), (1, -3/2*a - 1/2),
(1, -1/2*a - 1/2), (1, 1/2*a - 1/2), (1, 3/2*a - 1/2), (1, -a), (1, 0), (1, a), (1, 2*a),
(1, -1/2*a + 1/2), (1, 1/2*a + 1/2), (1, 3/2*a + 1/2), (1, 5/2*a + 1/2), (1, 1), (1, a + 1),
(1, 2*a + 1), (1, 3*a + 1), (a + 1, -3/2*a - 1/2), (a + 1, -1/2*a - 1/2),
(a + 1, 1/2*a - 1/2), (a + 1, 3/2*a - 1/2), (a + 1, -a), (a + 1, a), (a + 1, -1/2*a + 1/2),
(a + 1, 1/2*a + 1/2), (a + 1, 3/2*a + 1/2), (a + 1, 5/2*a + 1/2), (a + 1, 1),
(a + 1, 2*a + 1), (2*a + 1, -3/2*a - 1/2), (2*a + 1, -1/2*a - 1/2), (2*a + 1, 1/2*a - 1/2),
(2*a + 1, 3/2*a - 1/2), (2*a + 1, -a), (2*a + 1, 0), (2*a + 1, a), (2*a + 1, 2*a),
(2*a + 1, -1/2*a + 1/2), (2*a + 1, 1/2*a + 1/2), (2*a + 1, 3/2*a + 1/2),
(2*a + 1, 5/2*a + 1/2), (2*a + 1, 1), (2*a + 1, a + 1), (2*a + 1, 2*a + 1),
(2*a + 1, 3*a + 1), (3*a + 1, -3/2*a - 1/2), (3*a + 1, -1/2*a - 1/2), (3*a + 1, 1/2*a - 1/2),
(3*a + 1, 3/2*a - 1/2), (3*a + 1, -a), (3*a + 1, a), (3*a + 1, -1/2*a + 1/2),
(3*a + 1, 1/2*a + 1/2), (3*a + 1, 3/2*a + 1/2), (3*a + 1, 5/2*a + 1/2),
(3*a + 1, 1), (3*a + 1, 2*a + 1)]


"""
x = QQ['x'].0
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()
```

```
assert I.is_integral() == True
assert I != K.ideal(0)

Res = List_of_Residues(I)
U = Unit_Residues(I)

SET = []

for i in Res:
    for j in Res:
        SET.append((i,j))

for i in Res:
    for j in Res:
        if U.count(Residue_mod_Ideal  (((K.ideal(i) + K.ideal(j)).gens_two())[0],I)) == 0:
            if SET.count((i,j)) > 0:
                SET.remove((i,j))
return SET
```

```
###################################
## This section of code is meant##
## to build the necessary       ##
## commands for computing       ##
## Brandt matrices.             ##
###################################
```



```
def Brandt_Matrix(J,I):
    """
    We are working towards coding the action of the Hecke operator equivalent.
    This assume trivial character, and the level of the Hilbert modular forms
    space to be I. This will return the J^th Brandt matrix--where J is
    a totally positive generator of a prime ideal.
    =============
    EXAMPLES:
    =============
```

```
sage: Brandt_Matrix(K(3), K.ideal(4))
[10.0000000000000]
==============
sage: Brandt_Matrix(K(2), K.ideal(3))
[5.00000000000000]
==============
sage: Brandt_Matrix(K(2), K.ideal(5+2*(1+a)/2))
[1.00000000000000 4.00000000000000]
[4.00000000000000 1.00000000000000]
==============
sage: Brandt_Matrix(K((5+a/2)), K.ideal(5+2*(1+a)/2))
Traceback (most recent call last):
...
AssertionError
==============
sage: Brandt_Matrix(K(3), K.ideal(5+2*(1+a)/2))
[4.00000000000000 6.00000000000000]
[2.00000000000000 8.00000000000000]
==============
"""
x = QQ['x'].0
```

```
K.<a> = NumberField(x^2-5,embedding=1)
OK = K.ring_of_integers()

assert I.is_integral() == True
assert I != K.ideal(0)
assert J.is_totally_positive() == True

assert K.ideal(J).is_prime() == True



FD = Fundamental_Domain_Orbits(I)



TP = Theta_p(J)
Points=[[] for k in FD]
r = len(FD)
for k in FD:
    for i in TP:
        Points[FD.index(k)].append(
        Projective_Space_Representative((i[0][0]*k[0][0]+i[0][1]*k[0][1],
        i[0][2]*k[0][0]+i[0][3]*k[0][1]),I))
```

```
PreMatrix = []
for i in range(len(Points)):
    IM = [0 for j in range(len(FD))]
    for k in Points[i]:
        for j in FD:
            if j.count(k) >0:
                IM[FD.index(j)] += 1
    PreMatrix.append(IM)
Brandt = Matrix(CC, PreMatrix)
return Brandt
```