PERIODS AND SELMER GROUPS ASSOCIATED TO MOD p GALOIS REPRESENTATIONS OVER IMAGINARY QUADRATIC FIELDS

Lewis Combes

A thesis submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy
October 2023

Supervised by Haluk Şengün

University of Sheffield
Faculty of Science
School of Mathematics and Statistics
Pure Mathematics

Abstract

We develop methods for computing arithmetic invariants associated to mod p Galois representations over imaginary quadratic fields. These invariants (period polynomials, Selmer groups) are conjectured to fit into the classical landscape of the Langlands program in analogous ways. In particular, the vanishing of coefficient(s) of the period polynomial should capture the non-vanishing of an associated Selmer group.

Acknowledgements

First and foremost, I am indebted to my supervisor, Haluk Şengün, without whom none of this would have happened. In the last four years he has been an endless wellspring of enthusiasm, insight, energy and support. His indefatigable commitment to my mathematical development has been instrumental in all of my achievements. I am deeply grateful.

I must also thank my mum, Gill, and my brother, Alex, who mean the world to me and have given me so much. Likewise, my partner Tom has always been there when this work has seemed impassable. I love you all.

The friends I have made these last four years in Sheffield have been nothing but the best. Joseph, Kieran, David (and David), Jake, James; thank you all for allowing me to distract you from *your* PhDs when *my* PhD was frustrating.

I am grateful to the IT technicians here in Sheffield, and the team at the University of Sydney who maintain Magma. I have spent a great deal of time coding, and it would have been impossible without these people. In particular, John Cannon and Nicole Sutherland of Magma have been generous with their time and insights.

I am grateful to Alexandru Popa for sharing code to compute period polynomials of classical modular forms, and for Chris Williams, John Cremona and Aurel Page for helpful comments on the paper *Bianchi period polynomials: Hecke action and congruences* (the content of which appears throughout this thesis).

This work was completed while in receipt of the Engineering and Physical Sciences Research Council grant EP/R513313/1. I am grateful to EPSRC, and the public of the United Kingdom in general, for funding my studies.

Notation

K	A number field.
F	An extension of K.
L	A Galois extension of K, cut out by the image of a Galois representation.
$\mathcal{O}_E$	The ring of integers of a number field E.
p	Rational prime, usually the characteristic of a representation.
ρ	A Galois representation.
$[g, h]$	The commutator $g^{-1}h^{-1}gh$ of group elements g, h.
M^G	For M a G-module, the submodule of the G-fixed elements of M.
$A \mid_X$	The restriction of A to X.
$\text{rank}(E)$	The rank of the elliptic curve E (coefficient field usually implicit).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background

Let E be an elliptic curve over a number field K. The celebrated theorem of Mordell and Weil tells us that the abelian group of K-points of E, written $E(K)$, is finitely generated. Thus it can be written

$$E(K) \simeq T \oplus \mathbb{Z}^r$$

where T is a finite subgroup of torsion elements, and $\mathbb{Z}^r$ is free. Over a given number field, one has a good understanding of T; it is the value r, known as the **rank** of E, that is much more mysterious and has attracted a great deal of study in modern number theory.

One approach to study the rank of E, introduced by Cassells [Cas62], is the **Selmer group** (named in honour of Ernst Selmer). Writing $E[m]$ for the group of m-torsion points in $E(\overline{K})$, the m-Selmer group of E is defined (per Section X.4 of [Sil09]) as

$$\mathrm{Sel}_m(E/K) = \ker\left(H^1(\mathrm{Gal}(\overline{K}/K), E[m]) \to \prod_{\mathfrak{p}} H^1(G_{\mathfrak{p}}, E(\overline{K}))\right)$$

where $G_{\mathfrak{p}}$ is the decomposition group of $G_K = \mathrm{Gal}(\overline{K}/K)$ at $\mathfrak{p}$, and the product is taken over all places $\mathfrak{p}$ of K. See Chapter 2 for details on these constructions.

The utility of the Selmer group is that it lies in an exact sequence with other quantities of interest coming from E: that is, we have

$$0 \to E(K)/mE(K) \to \mathrm{Sel}_m(E/K) \to Ш(E/K) \to 0.$$

Here $Ш(E/K)$ is the Shaferavich-Tate group, defined also in Section X.4 of [Sil09]. If $E(K) \simeq T \oplus \mathbb{Z}^r$, then

$$E(K)/mE(K) \simeq T/mT \oplus (\mathbb{Z}/m\mathbb{Z})^r,$$

and so the rank of E can be computed from the rank of $\mathrm{Sel}_m(E/K)$, assuming sufficient knowledge about T and $Ш(E/K)$. In fact, the Shaferavich-Tate group is deeply mysterious; it is conjectured to be finite, but very little is concretely known. Assuming its order *can* be computed, finding the rank of E for a given curve becomes tractable by using the Selmer group.

The rank of E is also predicted to be connected to another deep arithmetic object, namely the L-function of E. Writing $\#E(\mathbb{F}_\mathfrak{p})$ for the number of points on E over a finite field $\mathbb{F}_\mathfrak{p} \simeq \mathcal{O}_K/\mathfrak{p}$ for $\mathfrak{p}$ a prime ideal of the ring of integers $\mathcal{O}_K$ of K, one can define the quantity

$$a_\mathfrak{p} \coloneqq 1 + N(\mathfrak{p}) - \#E(\mathbb{F}_\mathfrak{p}).$$

The **(Hecke)** L**-series** attached to E is the function of the complex variable s given by

$$L(E,s) = \prod_\mathfrak{p} L_\mathfrak{p}(E,s)^{-1},$$

where the **local Euler factors** $L_\mathfrak{p}$ is defined as

$$L_\mathfrak{p}(E,s) = 1 - a_p N(\mathfrak{p})^{-s} + N(\mathfrak{p})^{1-2s}$$

when $\mathfrak{p}$ does not divide the conductor of E (see see Section VIII.11 of [Sil09]); for the finite number of primes that do, more precise definitions of the local Euler factors are needed, see Appendix C.16 of [Sil09] for details. An important aspect of the theory of the elliptic curve L-series is that it should have a "special value" at $s = 1$. In fact, the product above only converges for $\mathrm{Re}(s) > \frac{3}{2}$, but it is conjectured that the L-series of an elliptic curve should have an analytic continuation to all of $\mathbb{C}$. This is known in general for $K = \mathbb{Q}$, and follows when E is modular, meaning there is a suitably defined modular form over K whose L-series matches that of E. See e.g. Section 5.9 of [DS16] for details when $K = \mathbb{Q}$.

The special value $L(E, 1)$ is predicted to vanish with multiplicity equal to the rank of E. This is the weak form of the *Birch & Swinnerton-Dyer conjecture* (**BSD**):

Conjecture 1.1.1 (Weak BSD)**.** Let E be an elliptic curve as above. Then $L(E, s)$ has a Taylor series around $s = 1$, with

$$L(E, s) = c(s - 1)^r + O((s - 1)^{r+1}),$$

where $r = \mathrm{rank}(E)$.

The strong form of **BSD** gives an exact expression for the coefficient c in terms of invariants of E and K.

Since its proposition, **BSD** has influenced a more general pattern of conjectures in number theory, for which evidence continues to build. The general concept is as follows: given an arithmetic object, the vanishing value of an L-function should detect the non-vanishing of a Selmer group, appropriately defined. The broadest version of this conjecture is the *Bloch-Kato conjecture* (**BK**). This is applied to a p-adic representation ρ of G_K, and states exactly this idea: there is a notion of an L-function and a Selmer group attached to ρ such that the critical L-value vanishes exactly when the Selmer group has non-zero rank. This reflects the classical case of an elliptic curve, up to some contribution to $\mathrm{Sel}_m(E/K)$ from the m-torsion in T.

Since we are not interested in the specifics of the Bloch-Kato conjecture, or p-adic Galois representations, we will not make explicit much of the theory—the interested reader should consult [Bel09].

The purpose of this thesis is to apply this philosophy to mod p Galois representations.

1.2 This thesis

The work in this thesis was initiated to refine and explore a question of Calegari and Venkatesh, posed in Section 10.3 of [CV12], which asks

Question 1.2.1. *Do periods of torsion classes detect classes in Galois cohomology?*

The authors go on to speculate that such a relationship should hold. We interpret the question as follows.

Let K be an imaginary quadratic field, with $\Gamma = \mathrm{PSL}_2(\mathcal{O}_K)$. There is an infinite family of commuting operators $\mathbb{T}_\mathfrak{n} = \{T_\mathfrak{p} \mid \mathfrak{p} \nmid \mathfrak{n}\}$ acting on $H^1(\Gamma_0(\mathfrak{n}), V(\mathbb{C}))$ called **Hecke operators**. Here $V = V_{k,l}^{s,t,\chi}$ is a weight module as defined in Section 4.1.2, with $V_{0,0}^{0,0,1}(\mathbb{C}) \simeq \mathbb{C}$. These Hecke operators have simultaneous eigenvectors, giving rise to *eigenvalue systems*. Galois representations are (conjecturally) connected to these eigenvalue systems in the following way: for a Galois representation ρ, one expects there to be a Hecke eigenvalue system $\{a_\mathfrak{p} : \mathfrak{p} \nmid \mathfrak{n}\}$ such that

$$a_\mathfrak{p} = \mathrm{trace}(\rho(\mathrm{Frob}_\mathfrak{p})),$$

where $\mathrm{Frob}_\mathfrak{p}$ is a Frobenius element, as defined in Section 2.1.1.

For modular elliptic curves E/K, there exist classes $[f]$ in $H^1(\Gamma_0(\mathfrak{n}), \mathbb{C})$, where $\mathfrak{n}$ is the conductor of E (see Example 2.2.6) and $\Gamma_0(\mathfrak{n})$ is a congruence subgroup (see Section 4.1.1), such that $L(E, 1)$ can be computed by evaluations of the **Kronecker pairing**

$$H^1(\Gamma_0(\mathfrak{n}), \mathbb{C}) \times H_1(\Gamma_0(\mathfrak{n}), \mathbb{C}) \to \mathbb{C},$$

given by

$$([f], [\gamma]) \mapsto f(\gamma).$$

Here we are interpreting f as a homomorphism $\Gamma_0(\mathfrak{n}) \to \mathbb{C}$, and γ as a matrix in $\Gamma_0(\mathfrak{n})$. Such an evaluation is called a **period** of f. The particular expression giving $L(E, 1)$ is given in Section 5.3.1.

Since we care about mod p Galois representations, we want to compute eigenvalue systems[1] in $H^1(\Gamma_0(\mathfrak{n}), V(\mathbb{F}_\mathfrak{p}))$ for prime ideals $\mathfrak{p}$ of K, called a **mod p eigenvalue system**. An eigenvalue system in $H^1(\Gamma_0(\mathfrak{n}), V(\mathbb{C}))$ is comprised of a series of algebraic integers, and their reductions modulo p appear as an eigenvalue system in $H^1(\Gamma_0(\mathfrak{n}), V(\mathbb{F}_\mathfrak{p}))$.

However, not every mod p eigenvalue system must arise this way. One aspect of the theory of Bianchi groups is the presence of torsion classes in $H^2(\Gamma_0(\mathfrak{n}), V(\mathcal{O}_K))$, which can prevent the lifting to characteristic 0 of mod p eigenvalue systems. The torsion classes mentioned in the question of Calegari and Venkatesh correspond to these non-lifting mod p eigenvalue systems.

The non-lifting eigenvalue systems in $H^1(\Gamma, V(\mathbb{F}_\mathfrak{p}))$ are mysterious objects, only recently proven to have associated Galois representations, in 2015 by Scholze [Sch15].

The construction in Section 5.3.1 used to extract information about $L(E, 1)$ from a class $[f] \in H^1(\Gamma_0(\mathfrak{n}), \mathbb{C})$ can be performed just as well on a class $[g] \in H^1(\Gamma_0(\mathfrak{n}), \mathbb{F}_\mathfrak{p})$, although the mod p computation loses its direct connection to the L-series of E, even when performed on the class whose Hecke eigenvalues are the mod p reduction of a class in $H^1(\Gamma_0(\mathfrak{n}), \mathbb{C})$. When performed on a class not lifting to characteristic 0, its meaning is even less clear, as there is not an L-series one can obviously associate to the class.

However, per Question 1.2.1, one expects the vanishing (or not) of this value for a class $[f]$ to behave very much like the vanishing (or not) of $L(E, 1)$—there should be some group attached to $[f]$ which is rank 0 when the period value is non-zero, and have positive rank when it is zero. This group should be a Selmer group, matching the broad theme described above. This is exactly what Question 1.2.1 means when it refers to "Galois cohomology classes"—although this description is slightly backwards; a more refined view of the premise starts with a mod p Galois representation ρ (a necessary ingredient for defining the Selmer group) and associates both a class in

[1] The Hecke operators acting on $H^1(\Gamma_0(\mathfrak{n}), V(\mathbb{F}_\mathfrak{p}))$ are not simultaneously diagonalisable in general. One must settle for the slightly weaker so-called *generalised eigenvectors*.

$H^1(\Gamma_0(\mathfrak{n}), V(\mathbb{F}_\mathfrak{p}))$ and a Selmer group, in the hopes that the two are compatible in this way. We will refer to this hypothetical compatibility as the **period-rank** relationship.

The purpose of this thesis, then, is to explore the properties of mod p Galois representations over K, hoping to define appropriately a period of the (conjecturally) associated class in mod p cohomology, and a Selmer group, such that the two satisfy the period-rank relationship. Throughout we will occasionally make reference to a hypothetical "correct" notion of the period and Selmer group, one that should satisfy this relationship (in a non-trivial fashion).

In the process of exploring this question, we have come to believe the appropriate quantity taking the role of the period should be some coefficient (or combination thereof) of the *period polynomial* of a class in $H^1(\Gamma, V(\mathbb{F}_\mathfrak{p}))$. Computing period polynomials associated to a particular Hecke eigenclass requires an action of Hecke matrices on the space of period polynomials, which we develop in Chapter 5. In doing so, we also find some interesting results on period polynomials associated to characteristic 0 classes, using them to detect and prove new congruences between Bianchi modular forms.

On the Selmer side, we develop algorithms to compute Selmer groups associated to a large class of representations, the so-called *nearly-ordinary representations*[2], which arise naturally from considering filtrations on the image of a Galois representation. Selmer groups such as the nearly-ordinary Selmer group are arithmetically interesting objects in their own right, often studied absent their possible connections to automorphic forms. With this in mind, we perform some arithmetic-statistical computations with mod 2 Selmer groups over $\mathbb{Q}$ and five imaginary quadratic fields.

[2] It should be noted, this terminology is not yet standardised, and the phrase "nearly-ordinary" can be used to mean several different things. For our exact formulation, see Chapter 2.

1.3 Structure of the thesis

In Chapter 2 of this thesis, we cover some important aspects of infinite Galois theory, mod p Galois representations ρ over a field K, and invariants of these representations that are used to associate to a class in the mod p cohomology of Γ to ρ. None of the content of this chapter is original.

In Chapter 3, we define group cohomology and use it to define Selmer groups attached to representations ρ. The general theory of Selmer groups we use is built around mirroring the constructions for p-adic Galois representations used to define, for example, the Selmer group appearing in **BK**, following standard constructions such as those in e.g. [Rub00]. We then use a connection between Galois cohomology and class field theory to derive methods for computing our Selmer groups explicitly. The derivation of this method and its implementation are original.

In Chapter 4, we record many useful results on the computation of the cohomology of $\Gamma = \mathrm{PSL}_2(\mathcal{O}_K)$ for five choices of imaginary quadratic field K. These results are well-known and none are original.

In Chapter 5, we prove facts about spaces of *period polynomials* associated to Γ, in the manner of [Kar22], [Wil17]. The construction of these spaces is not original; our contribution is to define a Hecke action there and prove it is compatible with the standard notion of Hecke operators on H^1 and H^2. This allows us to compute period polynomials associated to Bianchi modular forms in characteristic 0. This work was originally initiated to obtain information about period polynomials in characteristic p, however as we explored the situation in the (relatively) simpler case of characteristic 0, we found methods to use period polynomials to detect congruences of Bianchi modular forms. This chapter (and results quoted in the previous chapter) form the basis of the paper *Bianchi period polynomials: Hecke action and congruences* [Com23] (under review at time of submission, now accepted for publication *Research in Number Theory*). The only portion of this chapter not appearing in that paper is Section 5.3, which gives some philosophical justification for the kinds of choices of period one might wish to make when trying to extract arithmetic information from a Bianchi period polynomial.

In Chapter 6, we bring together the computational tools of Chapters 3, 4 and 5 to study some examples of Galois representations over K, computing Selmer groups and period polynomials. We also report on arithmetic-statistical studies of the average ranks of our Selmer groups over various fields, and study the finer points of ramification in nearly-ordinary Selmer groups as an attempt to detect to the hypothetical "correct" Selmer rank. This Chapter contains all original work, using data from the LMFDB [LMF23].

In the Appendices, we give all of the code used to compute the various quantities of interest. This comprises more than half the length of the text, although this is only in the interest of completeness. The reader can also find all the code on GitHub.

The code to compute the first cohomology of Bianchi groups can be found in the repository `BianchiFirstCoho`, at

https://github.com/lewismcombes/BianchiFirstCoho.

The version used to compute in this thesis is commit `4888a34`.

The code to compute period polynomials of Bianchi modular forms can be found in the repository `BianchiPeriodPols`, at

https://github.com/lewismcombes/BianchiPeriodPols.

The version used to compute in this thesis is commit `9726bf3`.

The code used to compute mod p Selmer groups can be found in the repository `ModpSelmerGroups`, at

https://github.com/lewismcombes/ModpSelmerGroups.

The version used to compute in this thesis is commit `7a0c614`.

1.4 Future work

Though we made progress in this thesis towards a precise statement of the conjectured period-rank relationship, we were limited somewhat by computational bottlenecks. In particular, the class field theory computations required to compute Selmer groups are expensive, and are not feasible to run on reasonable timescales for $|\mathrm{im}(\rho)|$ bigger than about 30.

There are also theoretical hurdles to pinning down the exact right period. Period polynomials of mod p representations need not have a central coefficient (in analogy with the classical case, see Section 5.3), and there are often several polynomials attached to the same representation, coming from different weights. The present iteration of the Selmer group computation doesn't take account of the weight; if it did, it should likely be in the form of a mod p version of the Tate twist (i.e. twisting V by the mod p cyclotomic character). If this is the case, the computation which makes no use of the Tate twist should correspond to the trivial weight system, but the images of these representations are generically big (at least all of $\mathrm{SL}_2(\mathbb{F}_p)$), and so are computationally very costly.

In future work, we hope to encode mod p Tate twists to allow different weights to be incorporated into Selmer computations. The development of speedups for these computations would also allow larger images to be tackled. Finally, further refinement of local conditions over p is expected to lead to a more robust Selmer group that could capture fully the conjectured relationship between periods and ranks.

2. GALOIS REPRESENTATIONS

2.1 Basics of infinite Galois theory

In order to make notions of infinite Galois theory precise, we will define profinite groups, using inverse limits. We follow [Koc02].

An **inverse system of groups** is a family $\mathcal{G} = (G_i)_{i \in I}$ of groups, indexed by some directed[1] poset I, and a family of homomorphisms $\Phi = (\phi_{i,j} : G_j \to G_i)_{i \le j}$, satisfying

1. $\phi_{i,i} : G_i \to G_i$ is the identity map,
2. $\phi_{i,k} = \phi_{i,j} \circ \phi_{j,k}$ for all $i \le j \le k$.

The **inverse limit of** $(\mathcal{G}, \Phi)$ is the subset of the direct product of the A_i given by

$$\varprojlim_{i \in I}(\mathcal{G}, \Phi) = \left\{ (g_i) \in \prod_i G_i \;\middle|\; \phi_{i,j}(g_j) = g_i \text{ for all } i \le j \right\}.$$

Informally, to define an element of the inverse limit, one needs to specify an element of each of the G_i, chosen in such a way as to be "compatible" with the maps $\phi_{i,j}$. A **profinite group** is the inverse limit of an inverse system of finite groups, where each finite group is equipped with the discrete topology. It is a **topological group**, meaning a group equipped with a topology such that multiplication and inversion are continuous.

[1] Meaning any pair of elements $i, j \in I$ has a shared upper bound k: $i \le k$, $j \le k$.

Example 2.1.1. The p-adic integers can be defined using an inverse limit. Using the directed poset $\mathbb{N}$ with the usual ordering, set $G_i = \mathbb{Z}/p^i\mathbb{Z}$, and choose the morphism $\phi_{i,j}$ to be the "reduction mod p^i" map $\mathbb{Z}/p^j\mathbb{Z} \to \mathbb{Z}/p^i\mathbb{Z}$. Then

$$\mathbb{Z}_p = \varprojlim_{n\in\mathbb{N}}(\mathcal{G}, \Phi).$$

Using the theory of profinite groups, one can also construct the absolute Galois group of a finite field $\mathbb{F}_p$:

Example 2.1.2. The finite Galois extensions of $\mathbb{F}_p$ are exactly the fields $\mathbb{F}_{p^n}$, with $\mathbb{F}_{p^i} \subseteq \mathbb{F}_{p^j}$ if and only if $i \mid j$. This gives the required partial ordering, coming from the divisibility ordering on $\mathbb{N}$. The groups $\mathrm{Gal}(\mathbb{F}_{p^n}/\mathbb{F}_p)$ are cyclic of order n, generated by the Frobenius element $\mathrm{Frob}_p : x \mapsto x^p$, of order n. The restriction $\mathrm{Gal}(\mathbb{F}_{p^j}/\mathbb{F}_p) \to \mathrm{Gal}(\mathbb{F}_{p^j}/\mathbb{F}_p)$ sends Frob_p to itself, so the restriction maps are

$$\phi_{i,j}(\sigma) = \sigma \pmod{i}.$$

The limit of this inverse system is $\widehat{\mathbb{Z}}$, the **profinite completion of** $\mathbb{Z}$.

The next example is the most important for us.

Example 2.1.3. Let K be a number field, and I the set of finite Galois extensions of K, ordered by inclusion. This is a directed set since any two fields $F_1, F_2 \in I$ extending K are both a subset of their compositum F_1F_2, also a finite Galois extension of K. Further, for $F_1 \subset F_2$, we have restriction maps sending $\sigma \in \mathrm{Gal}(F_2/K)$ to $\sigma\mid_{F_1} \in \mathrm{Gal}(F_1/K)$, so we get an inverse system of finite groups. The inverse limit of the system is the **absolute Galois group** of K, written $G_K = \mathrm{Gal}(\overline{K}/K)$.

Remark 2.1.4. Here $\overline{K}$ is a choice of algebraic closure of K. In general, the algebraic closure of K is only defined up to an K-automorphism, but up to that automorphism it is unique. We will study $\mathrm{Gal}(\overline{K}/K)$ via its representations, so this choice will not matter.

As G_K is a topological group, it is natural to ask exactly what the topology looks like.

Definition 2.1.5. We define the **Krull topology** on G_K by giving a base $\mathcal{B}$ of open neighbourhoods of the identity element:

$$\mathcal{B} = \left\{ \mathrm{Gal}(\overline{K}/F) \mid F/K \text{ finite} \right\}.$$

These are the normal subgroups of $\mathrm{Gal}(\overline{K}/K)$ of finite index. A base of open neighbourhoods for g is then given by $g\mathcal{B}$.

Alternatively, one can define the Krull topology by giving each finite Galois group the discrete topology, giving $\prod \mathrm{Gal}(F/K)$ the product topology, then giving $\mathrm{Gal}(\overline{K}/K) \subset \prod \mathrm{Gal}(F/K)$ the subspace topology. These two definitions are equivalent.

Under the Krull topology, two elements $\sigma, \tau \in G_K$ are "close" if they agree on a large finite extension F/K. It is used to define a modified version of the Galois correspondence for finite extensions: there is a bijection between *closed* subgroups $H \leq G_K$ and intermediate Galois extensions $\overline{K}/F/K$ given by

$$H \mapsto \overline{K}^H, \quad F \mapsto \mathrm{Gal}(\overline{K}/F). \tag{2.1}$$

2.1.1 Infinite Galois theory

Here we summarise some important aspects of infinite Galois theory over number fields, continuing to follow [Koc02].

We fix once and for all a choice of algebraic closure $\overline{K}$ of K.

Definition 2.1.6. An **absolute value** on K is a function $|\cdot| \colon F \to \mathbb{R}$ satisfying

1. Non-negativity: $|x| \geq 0$ for all $x \in K$;
2. Positive-definiteness: $|x| = 0$ if and only if $x = 0$;
3. Multiplicativity: $|xy| = |x| \cdot |y|$ for all $x, y \in K$;
4. Triangle inequality: $|x + y| \leq |x| + |y|$ for all $x, y \in K$.

Definition 2.1.7. An absolute value is **trivial** if it sends everything to 0. Two absolute values $|\cdot|_1, |\cdot|_2$ on K are **equivalent** if $|\cdot|_1 = |\cdot|_2^r$, for some real $r > 0$. Alternatively, an absolute value defines a metric by $d(x, y) = |x-y|$, and two absolute values are equivalent if their metrics define the same open sets.

Definition 2.1.8. A **place** $\mathfrak{p}$ of K is an equivalence class of non-trivial absolute values on K.

Remark 2.1.9. There is a bijection

$$\{\text{places of } K\} \leftrightarrow \{\text{prime ideals of } K\} \cup \left\{ \begin{array}{c} \text{embeddings of } K \\ \text{up to complex conjugation} \end{array} \right\}.$$

The places of K corresponding to prime ideals $\mathfrak{p}$ are called **finite places**, and their associated absolute values give rise to $\mathfrak{p}$-adic completions of K. The places corresponding to embeddings of K (either a real embedding, or a pair of conjugate complex embeddings) give rise to $\mathbb{R}$ or $\mathbb{C}$ as completions of K, and are called **real** or **complex** accordingly.

Remark 2.1.10. We will use the same notation $\mathfrak{p}$ to refer to a place and its corresponding ideal in K when $\mathfrak{p}$ is finite. When $\mathfrak{p}$ is infinite, there is no corresponding ideal, but we will still write it as $\mathfrak{p}$ for uniformity.

Since $\mathfrak{p}$ defines an absolute value on K, we can define the field $K_{\mathfrak{p}}$ as the completion of K with respect to $\mathfrak{p}$. When $K = \mathbb{Q}$, these completions are $\mathbb{R}$ (corresponding to the usual absolute value on $\mathbb{Q}$, coming from the embedding $\mathbb{Q} \hookrightarrow \mathbb{R}$), and the fields $\mathbb{Q}_p$ for all primes p. For a general number field K, the completions are $\mathbb{R}$ or $\mathbb{C}$, and finite extensions of $\mathbb{Q}_p$.

The notion of a place extends naturally to $\overline{K}$, where it is again an equivalence class of non-trivial absolute values. For an extension F/K, we say a place $\mathfrak{q}$ of F **extends** $\mathfrak{p}$ if $\mathfrak{q} \mid_K = \mathfrak{p}$.

We fix a place $\mathfrak{P}$ of $\overline{K}$ that extends $\mathfrak{p}$. This has the effect of fixing a place $\mathfrak{q}$ of F that extends $\mathfrak{p}$ for all finite Galois F/K: the place $\mathfrak{q}$ is exactly $\mathfrak{P} \mid_F$. These choices are all compatible, in the sense that, if $F_1/F_2/K$ is a tower of fields Galois over K, then $(\mathfrak{P}\mid_{F_1})\mid_{F_2} = \mathfrak{P}\mid_{F_2}$.

If we instead chose, for every finite Galois F/K, a place $\mathfrak{q}$ extending $\mathfrak{p}$ in a manner *compatible* with the inverse system[2] from Example 2.1.3, we would get a place $\mathfrak{P}$ of $\overline{K}$ that restricts to each place $\mathfrak{q}$ for F/K as above.

When $\mathfrak{p}$ is finite, the choice of an extension $\mathfrak{q}$ of $\mathfrak{p}$ to F/K can be written as ideals as $\mathfrak{q} \mid \mathfrak{p}$. So for a finite place $\mathfrak{p}$, the choice $\mathfrak{P}$ fixes prime ideals over $\mathfrak{p}$ for all finite Galois F/K.

The completion of $\overline{K}$ with respect to $\mathfrak{P}$ is $\overline{K}_{\mathfrak{P}} := \bigcup F_{\mathfrak{P}|_F}$, where the union runs over all finite Galois F/K. This gives a canonical embedding $\overline{K} \hookrightarrow \overline{K}_{\mathfrak{P}}$, allowing us to define decomposition and inertia groups at a place $\mathfrak{p}$ of K.

Definition 2.1.11. Let $\mathfrak{p}$ be a finite place of K, and $\mathfrak{P}$ a place of $\overline{K}$ extending $\mathfrak{p}$. The embedding $\overline{K} \hookrightarrow \overline{K}_{\mathfrak{P}}$ induces an embedding $G_{K_\mathfrak{p}} = \operatorname{Gal}(\overline{K}_{\mathfrak{P}}/K_\mathfrak{p}) \hookrightarrow G_K$. This $G_{K_\mathfrak{p}}$ is the **decomposition group of K at $\mathfrak{p}$**. When K is clear from context we will also write $G_{K_\mathfrak{p}} = D_\mathfrak{p}$.

Remark 2.1.12. The group $D_\mathfrak{p}$ is defined only up to the choice of place $\mathfrak{P}$ extending $\mathfrak{p}$. The group G_K acts transitively on these places (per e.g. Section 2 of the appendix to [Was96]), so $D_\mathfrak{p}$ is only defined in G_K up to conjugacy. Our main interest in these groups will be in the context of representations of G_K, where this is less of a problem. For practical purposes, we will find it sufficient to choose a particular extension of a place and compute with that particular decomposition subgroup.

For a finite Galois extension F/K, we can define the decomposition group alternately:

$$D_\mathfrak{p}(F/K) = \{\sigma \in \operatorname{Gal}(F/K) \mid \sigma(\mathfrak{q}) = \mathfrak{q}\}. \tag{2.2}$$

This group also depends on the ideal $\mathfrak{q}$, and is only defined up to conjugacy in $\operatorname{Gal}(F/K)$. Since this is determined by our choice of $\mathfrak{P}$, taking the inverse limit $\varprojlim D_\mathfrak{p}(F/K)$ recovers the decomposition group $\operatorname{Gal}(\overline{K}_{\mathfrak{P}}/K_\mathfrak{p})$. Each individual $D_\mathfrak{p}(F/K)$

[2] Meaning that, for two fields F_1, F_2 finite Galois over K with respective places $\mathfrak{q}_1, \mathfrak{q}_2$, the associated place extending $\mathfrak{p}$ of the compositum F_1F_2 we choose is exactly the one that restricts to $\mathfrak{q}_i$ on F_i.

is the Galois group of the p-adic extension $F_{\mathfrak{q}}/K_{\mathfrak{p}}$.

Next, we define the inertia subgroup of G_K at a place $\mathfrak{p}$.

Definition 2.1.13. We write $\mathbb{F}_{\mathfrak{p}}$ for the residue field of $K_{\mathfrak{p}}$; then the residue field of $\overline{K}_{\mathfrak{P}}$ is $\overline{\mathbb{F}_{\mathfrak{p}}}$, and there exists a surjection

$$\mathrm{Gal}(\overline{K}_{\mathfrak{P}}/K_{\mathfrak{p}}) \twoheadrightarrow \mathrm{Gal}(\overline{\mathbb{F}_{\mathfrak{p}}}/\mathbb{F}_{\mathfrak{p}}).$$

The **inertia subgroup of G_K at $\mathfrak{p}$**, denoted $I_{K_{\mathfrak{p}}}$ (or $I_{\mathfrak{p}}$ when K is clear from context) is the kernel of this map.

As with the decomposition group, the inertia group can be defined for a finite Galois extension F/K:

$$I_{\mathfrak{p}}(F/K) = \{\sigma \in \mathrm{Gal}(F/K) \mid \sigma(x) - x \in \mathfrak{q} \text{ for all } x \in \mathcal{O}_F\}.$$

This group is again only defined up to conjugacy in $\mathrm{Gal}(F/K)$. Since we use the same $\mathfrak{q}$ to define the decomposition and inertia subgroup, there is an inclusion $I_{\mathfrak{q}}(F/K) \leq D_{\mathfrak{q}}(F/K)$. Taking the inverse limit $\varprojlim I_{\mathfrak{q}}(F/K)$ over finite Galois extensions again yields the group $I_{\mathfrak{p}}$ and the inclusion $I_{\mathfrak{p}} \leq D_{\mathfrak{p}}$.

Remark 2.1.14. The decomposition and inertia groups $D_{\mathfrak{p}}(F/K)$ and $I_{\mathfrak{p}}(F/K)$ depend on the choice of place $\mathfrak{q}$ of L extending $\mathfrak{p}$. As we will usually work with a specific (implicit) choice of $\mathfrak{q}$, we will omit the choice from the notation. In the rare instances where we want to consider decomposition or inertia groups for a *particular choice* of $\mathfrak{q}$, we will write them as $D_{\mathfrak{p}}^{\mathfrak{q}}(F/K)$ and $I_{\mathfrak{p}}^{\mathfrak{q}}(F/K)$.

Definition 2.1.15. A place $\mathfrak{p}$ of K is **unramified** in a finite extension F/K if $I_{\mathfrak{p}}(F/K) = 1$, i.e. if the inertia subgroup is trivial. We also say that F/K is unramified at $\mathfrak{p}$. If F/K is infinite, $\mathfrak{p}$ is unramified if the inverse limit $\varprojlim I_w(F'/K)$ over finite Galois subextensions F'/K of F/K is trivial.

We will also use the notions of inertia and decomposition for infinite places, which are simpler.

Definition 2.1.16. We have the above inclusion

$$\mathrm{Gal}(\overline{K}_{\mathfrak{P}}/K_{\mathfrak{p}}) \hookrightarrow \mathrm{Gal}(\overline{K}/K).$$

For an infinite place $\mathfrak{p}$, we define $D_{\mathfrak{p}} = I_{\mathfrak{p}}$ to be the image of this map. For a finite extension F/K, the place $\mathfrak{p}$ is **unramified** if $F_{\mathfrak{q}} = K_{\mathfrak{p}}$.

An infinite place of K is either a real or a complex embedding. This definition says such a place $\mathfrak{p}$ is ramified in F/K if and only if $\mathfrak{p}$ is real in K, and extends only to complex places $\mathfrak{q}$ in F. Diagrammatically, we can represent this as

$$\begin{tikzcd} \mathbb{R} \arrow[r] \arrow[dr] & \mathbb{R} \\ \mathbb{C} \arrow[r] & \mathbb{C} \end{tikzcd}$$

with the diagonal arrow being the only possible ramification of $\mathfrak{p}$ when extended to $\mathfrak{q}$.

Within the inertia subgroup for a finite extension, there exist **higher ramification groups** we will need in Section 2.3.1.

Definition 2.1.17. Let $i \geq 0$. The i^{th} **ramification group of** F/K **at** $\mathfrak{p}$ is the subgroup of $\mathrm{Gal}(F/K)$ given by

$$G_{\mathfrak{p},i}(F/K) = \left\{\sigma \in \mathrm{Gal}(F/K) \mid \sigma(x) - x \in \mathfrak{q}^{i+1} \text{ for all } x \in \mathcal{O}_F\right\}.$$

We have $G_{\mathfrak{p},0}(F/K) = I_{\mathfrak{p}}(F/K)$.

Finally, we note some useful results from infinite Galois theory.

Proposition 2.1.18. Let F/K be a finite extension. Then

$$D_{\mathfrak{p}}(F/K) = \mathrm{Gal}(F/K) \cap D_{\mathfrak{p}},$$
$$I_{\mathfrak{p}}(F/K) = \mathrm{Gal}(F/K) \cap I_{\mathfrak{p}}.$$

2.2 Mod p Galois representations

We introduce mod p Galois representations and recap some important invariants in the 2-dimensional case.

Definition 2.2.1. A **mod p Galois representation** over K is a homomorphism

$$\rho\colon G_K \to \mathrm{GL}(V),$$

where V is a vector space over $\overline{\mathbb{F}}$, for $\mathbb{F}$ a finite field of characteristic p.

We will restrict our attention to *continuous* Galois representations, meaning ρ is continuous with respect to the Krull topology on G_K and the discrete topology on V. When ρ is continuous, the identity element in $\mathrm{GL}(V)$ is open, so its preimage (the kernel of ρ) must be open. It is a standard result of the theory of topological groups that open subgroups are also closed, so there is a finite Galois extension L/K such that $\ker(\rho)$ contains $\mathrm{Gal}(\overline{K}/L)$, and

$$L = \overline{K}^{\ker(\rho)}, \quad \ker(\rho) \simeq \mathrm{Gal}(\overline{K}/L)$$

by the Galois correspondence (2.1).

Definition 2.2.2. We call the above L the **splitting field** of ρ.

Basic group theory tells us that $\mathrm{im}(\rho) \simeq G_K/\ker(\rho)$. For a finite extension of number fields L/K, we always have that $\overline{L} = \overline{K}$, so $\mathrm{Gal}(\overline{K}/L) = \mathrm{Gal}(\overline{L}/L) =: G_L$; the Galois correspondence then tells us that

$$\mathrm{Gal}(L/K) \simeq \mathrm{im}(\rho) \simeq G_K/G_L. \tag{2.3}$$

We will use this identification frequently throughout Chapter 3.

Since $\mathrm{im}(\rho)$ is finite, we can realise $\mathrm{GL}(V)$ as a vector space over a finite field of characteristic p.

Definition 2.2.3. The dimension of a continuous mod p Galois representation is the dimension of V as an $\mathbb{F}_q$-vector space.

The local behaviour of a Galois representation is important for understanding and identifying it. To that end, we introduce the notions of ramification of representations, and Frobenius elements.

Definition 2.2.4. A Galois representation $\rho\colon G_K \to \mathrm{GL}(V)$ is **unramified at a place** $\mathfrak{p}$ if it is trivial on the inertia at $\mathfrak{p}$; i.e.

$$\rho(I_{\mathfrak{p}}) = 1.$$

Equivalently, ρ is unramified at $\mathfrak{p}$ if $I_{\mathfrak{p}}(L/K) = 1$. Since L/K is a finite Galois extension, it is only ramified at finitely many places, so ρ is unramified at all but finitely many places of L. We also say ρ is unramified **almost everywhere**[3].

Recall from Example 2.1.2 that $\mathrm{Gal}(\overline{\mathbb{F}_{\mathfrak{p}}}/\mathbb{F}_{\mathfrak{p}}) \simeq \widehat{\mathbb{Z}}$. We write $\mathrm{Frob}_{\mathfrak{p}}$ for a generator of $\mathrm{Gal}(\overline{\mathbb{F}_{\mathfrak{p}}}/\mathbb{F}_{\mathfrak{p}})$. The preimage of $\mathrm{Frob}_{\mathfrak{p}}$ is a coset of $I_{\mathfrak{p}}$ in $\mathrm{Gal}(\overline{K}_{\mathfrak{P}}/K_{\mathfrak{p}})$, and it is common to refer to any element of this coset as *a* Frobenius element at $\mathfrak{p}$, also denoted $\mathrm{Frob}_{\mathfrak{p}}$. If a representation ρ is unramified at $\mathfrak{p}$, then $\rho(\mathrm{Frob}_{\mathfrak{p}})$ is independent of this choice.

We now list some examples of Galois representations.

Example 2.2.5 (mod p cyclotomic character)**.** For a prime p, write μ_p for the group of the p^{th} roots of unity in $\mathbb{C}$. Choose a generator ζ of μ_p. The group $G_{\mathbb{Q}}$ acts on μ_p by automorphisms, so for $\sigma \in G_{\mathbb{Q}}$, we have

$$\zeta \cdot \sigma = \zeta^{k_\sigma},$$

for some $k_\sigma \in \mathbb{F}_p^\times$. The **mod p cyclotomic character** is the map $\chi_p\colon G_{\mathbb{Q}} \to \mathbb{F}_p^\times$ defined by $\sigma \mapsto k_\sigma$. It is a continuous 1-dimensional Galois representation, unramified for all primes $\ell \neq p$.

Example 2.2.6 (Mod p representation of an elliptic curve)**.** This example is the most important for our purposes, and will be used extensively in Chapter 6. Let E

[3] Here we are using "almost all" to mean "all but finitely many", i.e. "almost all with respect to the counting measure".

be an elliptic curve over K, that is, a smooth projective curve over K of genus 1 with a distinguished K-point $\mathcal{O}$. We write $E(F)$ for the set of F-points on E for fields F/K. The set $E(F)$ forms a group under the standard tangent-chord construction (see e.g. Chapter III of [Sil09]).

The curve E has a model over K

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

with the $a_i \in K$. The group G_K acts on $E(\overline{K})$ co-ordinate-wise, that is,

$$\sigma((x, y)) = (\sigma(x), \sigma(y))$$

for $\sigma \in G_K$, (x, y) on the affine part[4] of $E(\overline{K})$. The set $E[p]$ of points (x, y) in $E(\overline{K})$ such that $p \cdot (x, y) = 0$ is a subgroup of $E(\overline{K})$, since the addition operations for K are defined by K-rational functions and so are fixed by G_K. In fact, the group is finite:

$$E[p] \simeq \mathbb{F}_p \oplus \mathbb{F}_p.$$

This gives a continuous two-dimensional Galois representation of G_K, written $\rho_{E,p}$. The splitting field of $\rho_{E,p}$ is the result of adjoining the co-ordinates of every point in $E[p]$ to K, sometimes written $K(E[p])$. This field can be computed explicitly, which we will do for many later examples. We accomplish this with code written by Andrew Sutherland in [Sut16].

The **conductor** of E is an ideal $\mathfrak{n} \triangleleft \mathcal{O}_K$ divisible only by the primes of bad reduction of E, see Section VIII.11 of [Sil09]. The representation $\rho_{E,p}$ is unramified at primes $\mathfrak{q} \nmid \mathfrak{n}p$, and, for these primes, we have

$$\operatorname{trace}(\rho(\operatorname{Frob}_{\mathfrak{q}})) = N(\mathfrak{q}) + 1 - \#E(\mathbb{F}_{\mathfrak{q}}) =: a_{\mathfrak{q}}.$$

[4] The projective part can be shown to contain only one point, and when this is taken to be the identity of the group operation, it is fixed by the action of G_K.

2.3 Invariants

In order to compute "automorphic" information associated to Galois representations, we need a way to associate cohomology classes (see Chapter 4) to a given representation ρ. To do this, we define various invariants that (conjecturally) pin down exactly where to find cohomology classes that correspond to ρ, in the manner described in Section 4.4.

The specific pieces of information required are the Serre conductor, the Serre weight(s), the character, and the traces of Frobenius elements. From here we will assume that ρ is a 2-dimensional Galois representation over K an imaginary quadratic field.

We write $L = \overline{K}^{\ker(\rho)}$. In the following, we will make various references to concepts introduced in Chapter 4, indicating in which specific section the concept is used when we do so.

Our exposition here owes much to that of [Tor12], which we follow closely.

2.3.1 Serre conductor

The Serre conductor of a Galois representation over K is an ideal in $\mathcal{O}_K$, used to define the level of associated cohomology classes (Section 4.4).

Definition 2.3.1. Let $\mathfrak{p}$ be a prime of K, and write $G_{\mathfrak{p},i}$ for the i^{th} ramification group of ρ at $\mathfrak{p}$. Writing

$$V_{\mathfrak{p},i} = \{v \in V \mid \rho(g) \cdot v = v \text{ for all } g \in G_{\mathfrak{p},i}\}$$

for the subspace of V of points invariant under the action of $G_{\mathfrak{p},i}$, we define the **Serre exponent of ρ at $\mathfrak{p}$** as

$$e_{\mathfrak{p}}(\rho) = \sum_{i=0}^{\infty} \frac{1}{[G_{\mathfrak{p},0} : G_{\mathfrak{p},i}]} \dim(V/V_{\mathfrak{p},i}).$$

Since ramifications groups are eventually trivial, the dimension of $V/V_{\mathfrak{p},i}$ is eventually 0, so the sum is finite. When ρ is unramified at $\mathfrak{p}$, the Serre exponent is 0.

Definition 2.3.2. The **Serre conductor** of ρ is the ideal

$$\mathfrak{n}(\rho) = \prod_{\substack{\mathfrak{p} \text{ prime} \\ \mathfrak{p} \nmid p}} \mathfrak{p}^{e_{\mathfrak{p}}(\rho)}.$$

Since ρ is unramified almost everywhere, the product is over finitely many terms, with finite exponents $e_{\mathfrak{p}}(\rho)$, so this expression defines an ideal of K.

2.3.2 Serre weight(s)

A Serre weight attached to a representation tells one which weight module to use to find associated cohomology classes. When $K = \mathbb{Q}$, there is one Serre weight that is typically used, which is the minimal such weight. When K is an imaginary quadratic, there is no obvious notion of a minimal weight, so a representation ρ usually has several weights associated to it.

Definition 2.3.3. A **Serre weight** over K is an irreducible $\overline{\mathbb{F}_p}$-representation V of $\mathrm{GL}_2(\mathcal{O}_K/p\mathcal{O}_K)$.

Per [Tor12], for K an imaginary quadratic field the Serre weights are of the form

$$V = \bigotimes_{\mathfrak{p}|p} V_{\mathfrak{p}},$$

where p is the rational prime over which $\mathfrak{p}$ lies. Here, the $V_{\mathfrak{p}}$ are the modules

$$V_{\mathfrak{p}} = \bigotimes_{\tau \in S_{\mathfrak{p}}} (\det{}^{a_\tau} \otimes_{\mathbb{F}_{\mathfrak{p}}} \mathrm{Sym}^{k_\tau}(\mathbb{F}_{\mathfrak{p}}^2)) \otimes_\tau \overline{\mathbb{F}_p},$$

where $S_{\mathfrak{p}}$ is the set of embeddings $\tau \colon \mathbb{F}_{\mathfrak{p}} \hookrightarrow \overline{\mathbb{F}_p}$, and $\mathrm{Sym}^k(\mathbb{F}_{\mathfrak{p}}^2)$ is the k^{th} symmetric power of the standard representation on $\mathbb{F}_{\mathfrak{p}}^2$ of $\mathrm{GL}_2(\mathcal{O}/p\mathcal{O})$. The values a_τ and k_τ vary between 0 and $p-1$, as when $a_\tau \geq p$, the determinant power is just $a_\tau \pmod p$, and when $k_\tau \geq p$, the representation is no longer irreducible.

These modules are the $\mathbb{F}_\mathfrak{p}$-versions of the weight modules described in Section 4.1.2.

2.3.3 Character

The representation $\rho\colon G_K \to \mathrm{GL}_2(\mathbb{F}_q)$ has a character associated to it, first given by Serre in [Ser87] for representations of $G_\mathbb{Q}$—the same principle works in general. The determinant of ρ is a 1-dimensional Galois representation (i.e. a character)

$$\det \rho\colon G_K \to \mathbb{F}_q^\times.$$

We will define the character of ρ (also known as its *nebentype*) by removing the p-part of the determinant. More precisely, class field theory reinterprets $\det \rho$ as a character

$$\det \rho\colon (\mathcal{O}_K/p\mathfrak{n}(\rho))^\times \to \overline{\mathbb{F}_p}^\times,$$

which then factors as $\chi\varepsilon$, with

$$\chi\colon (\mathcal{O}_K/p\mathcal{O}_K)^\times \to \overline{\mathbb{F}_p}^\times, \quad \varepsilon\colon (\mathcal{O}_K/\mathfrak{n}(\rho))^\times \to \overline{\mathbb{F}_p}^\times.$$

We take ε to be the character of ρ.

2.3.4 Traces of Frobenius elements

For each prime $\mathfrak{p}$ of $\mathcal{O}_K$, we have a choice of Frobenius element in $D_\mathfrak{p}(L/K) \leq \mathrm{Gal}(L/K)$, where L is again the splitting field of ρ. When ρ is unramified, the trace of $\rho(\mathrm{Frob}_\mathfrak{p})$ does not depend on the choice, hence we take the traces of Frobenius for all primes $\mathfrak{p} \nmid p\mathfrak{n}(\rho)$ as an invariant.

Under the correspondence predicted by Serre's conjecture, the trace of Frobenius of ρ at a prime $\mathfrak{p} \nmid p\mathfrak{n}(\rho)$ should match the Hecke eigenvalue of the associated cohomology class at $\mathfrak{p}$. To identify such a class, we compute these traces up to some bound. This does not strictly prove the association, but until Serre's conjecture is proven this is considered sufficient for its verification.

3. SELMER GROUPS

In this chapter, we introduce Galois cohomology groups $H^1(G_K, V)$. Selmer groups are special subgroups of Galois cohomology groups, defined using local information arising from the completions $K_\mathfrak{p}$. We introduce three Selmer groups we can associate to mod p representations, and, using the inflation-restriction sequence in Galois cohomology, we translate the problem of computing a Selmer group into a problem in class field theory, which can then be solved using existing algorithms.

3.1 Group cohomology

We will first define group cohomology generally (as we will also need it for Chapter 4), before specifying to Galois cohomology. The theory of group cohomology can be developed abstractly from complexes and Ext functors, but we will only be using the explicit description of cohomology in terms of cocycles and coboundaries, so that is what we develop here. We follow the expositions of [Har20] and [Bro12], presenting the results we need without proof.

Let G be a group and M a right G-module—we will denote the action of $g \in G$ on $m \in M$ by $m \cdot g$. For $i \geq 0$, write A^i for the abelian group of functions $f : G^i \to M$, with the convention that $A^0 = M$, the space of functions $\{1\} \to M$. Define the **coboundary map** $d^i \colon A^i \to A^{i+1}$ as

$$\begin{aligned}(d^if)(g_1,\dots,g_{i+1}) =& f(g_1,\dots,g_i)\cdot g_{i+1} + \sum_{j=1}^{i}(-1)^j f(g_1,\dots,g_jg_{j+1},\dots,g_{i+1}) \\ &+(-1)^{i+1}f(g_2,\dots,g_{i+1}).\end{aligned}$$

Definition 3.1.1. The space of i-**cocycles** is

$$Z^i(G,M) \coloneqq \ker(d^i).$$

Definition 3.1.2. The space of i-**coboundaries** is

$$B^i(G,M) \coloneqq \operatorname{im}(d^{i-1}).$$

A simple calculation shows that $d^i \circ d^{i-1} = 0$, and so we have

$$B^i(G,M) \leq Z^i(G,M).$$

Definition 3.1.3. The i^{th} **cohomology group of G with coefficients in M** is the space

$$H^i(G,M) = \frac{Z^i(G,M)}{B^i(G,M)}.$$

The advantage of this formulation of group cohomology is that it can be very explicitly written down. For example, when $i = 0$, a quick calculation shows that

$$H^0(G,M) = M^G.$$

When $i = 1$, the definition of d^1 shows that $Z^1(G,M)$ is the space of functions $f\colon G \to M$ satisfying

$$f(gh) = f(g)\cdot h + f(h).$$

Such functions are sometimes called *crossed homomorphisms.* We will call this the

1-cocycle property. Meanwhile, the space $B^1(G, M)$ is the space of functions

$$f_m \colon g \mapsto m \cdot g - m$$

for some $m \in M$. These are sometimes called *principal crossed homomorphisms.* When the action of G on M is trivial, the 1-cocycle condition becomes $f(gh) = f(g) + f(h)$, and the 1-coboundary condition defines only the zero map, so we have that the cohomology group $H^1(G, M)$ is just $\mathrm{Hom}(G, M)$.

The description of second cohomology in terms of cocycles and coboundaries is a little less useful than first cohomology, though we still note it here for completion. The space $Z^2(G, M)$ is made up of all functions $f : G^2 \to M$ satisfying

$$f(g, h) \cdot k - f(gh, k) + f(g, hk) - f(h, k) = 0.$$

The space $B^2(G, M)$ is, in turn, made up of all functions $f : G^2 \to M$ such that there is a function $\psi : G \to M$ satisfying

$$f(g, h) = \psi(g) \cdot h - \psi(gh) + \psi(h).$$

One useful property of cohomology is that it turns short exact sequences into long exact sequences; we have the following.

Proposition 3.1.4. Let A, B, C be right G-modules with

$$0 \to A \to B \to C \to 0 \tag{3.1}$$

exact. Then there is an exact sequence of cohomology groups

$$\begin{aligned} 0 \to & H^0(G, A) \to H^0(G, B) \to H^0(G, C) \to \\ & H^1(G, A) \to H^1(G, B) \to H^1(G, C) \to \\ & H^2(G, A) \to \dots \end{aligned}$$

The maps between cohomology groups of the same degree are induced by the maps between the underlying modules, e.g. $H^1(G, A) \to H^1(G, B)$ comes from the map $\phi\colon A \to B$ in (3.1): the class $[f] \in H^1(G, A)$ is taken to $[\phi \circ f]$. We write the map between cohomology groups ϕ^*, and say it is **induced** by ϕ. The maps between groups of different degrees are called the **connecting homomorphisms**, and proving these exist and give rise to the exact sequence is the main content of the proof of the proposition.

If G, G' are groups with a homomorphism $\varphi\colon G' \to G$, and M is a G-module, we can make M a G'-module by setting

$$m \cdot g' \coloneqq m \cdot \varphi(g').$$

This gives rise to a homomorphism in cohomology

$$\varphi^*\colon H^i(G, M) \to H^i(G', M). \tag{3.2}$$

When $G' = H \leq G$, this map is called **restriction**. The analogue of Proposition 3.1.4 is the inflation-restriction sequence, which we describe below in Definition 3.1.10.

In Section 4.1.1, we will relate the cohomology of a subgroup $H \leq G$ to the cohomology of G itself. To do this, we introduce the induced and coinduced modules.

Definition 3.1.5. For $H \leq G$ and M an H-module (over a ring R), define the **induced module**

$$\mathrm{Ind}_H^G(M) = R[G] \otimes_{R[H]} M$$

and the **co-induced module**

$$\mathrm{CoInd}_H^G(M) = \mathrm{Hom}_{R[H]}(R[G], M).$$

Assuming $[G : H] < \infty$, we can write

$$G = \coprod_{i=1}^{n} Hr_i$$

where $\{r_i\}$ are representatives of $H\backslash G$, giving the decomposition

$$\mathrm{Ind}_H^G(M) = \bigoplus_{r_i \in H\backslash G} r_i \otimes M,$$

where $g \otimes M = \{g \otimes m \mid m \in M\}$. We can rewrite this as

$$\mathrm{Ind}_H^G(M) = R[H\backslash G] \otimes M.$$

Here $R[H\backslash G]$ has the right G-action induced by the permutation of $\{Hr_i\}$ by elements of G.

Remark 3.1.6. There is a lack of consensus about which way around the notions of induced and co-induced module ought to be. We have opted to follow the exposition of [Bro12] in this regard. As the following proposition shows, there is often little need for the distinction.

Proposition 3.1.7. When $[G : H] < \infty$, we have

$$\mathrm{Ind}_H^G(M) \simeq \mathrm{CoInd}_H^G(M).$$

To relate the cohomology of subgroups and their supergroups, we have the following result:

Proposition 3.1.8 (Shapiro's lemma)**.** The map $\psi\colon \mathrm{CoInd}_H^G(M) \to M$ sending f to $f(1)$ (equivalently, the coefficient of the coset H) is a homomorphism of modules compatible with the inclusion map $H \to G$. It gives rise to the map in cohomology

$$\psi^*\colon H^i(G, \mathrm{CoInd}_H^G(M)) \to H^i(H, M),$$

which is an isomorphism.

The proof is given as (6.6) in [Bro12]. We will only be using this result in the context of subgroups such that $[G : H] < \infty$. So, in particular, we will be using the modified version of Shapiro's lemma, with the isomorphism

$$\psi^*\colon H^i(G, R[H\backslash G] \otimes M) \to H^i(H, M).$$

Definition 3.1.9. A **Galois cohomology group** is a group $H^i(\mathrm{Gal}(F/K), M)$, where F/K is a Galois extension of number fields, restricted to only those (classes of) cocycles that are continuous with respect to the topologies on $\mathrm{Gal}(F/K)$ and M. As all our M will be of the form $\mathrm{GL}(V)$ with V a vector space over $\mathbb{F}_p$, we always take the discrete topology on M.

When F/K is finite, this coincides with the normal cohomology group above. When F/K is infinite, the continuity condition ensures we have the relation

$$H^i(\mathrm{Gal}(F/K), M) = \varprojlim H^i(\mathrm{Gal}(F'/K), M),$$

where the limit runs over all finite Galois F'/K, and M is made an $\mathrm{Gal}(F'/K)$-module by restriction.

Finally, we will also need the inflation-restriction sequence.

Definition 3.1.10. Let $N \leq G$ be a normal subgroup, and M a G-module. Then the sequence

$$0 \to H^1(G/N, M^N) \xrightarrow{\mathrm{inf}} H^1(G, M) \xrightarrow{\mathrm{res}} H^1(N, M)^{G/N} \to H^2(G/N, M^N) \to \dots$$

is exact. See [Wei94], Section 6.7.

The map $\mathrm{inf}\colon H^1(G/N, M^N) \to H^1(G, M)$ is called the **inflation** map, and is induced by the map $G^i \to (G/N)^i$, noting that $M^N \subset M$; the map $\mathrm{res}\colon H^1(G, M) \to H^1(N, M)^{G/N}$ is the **restriction** map, as in (3.2). The group G/N acts on $H^1(N, V)$ by conjugation: for $f \in H^1(N, M)$ and $g \in G/N$, we have

$$(g \cdot f)(n) = f(g^{-1}ng)g^{-1}.$$

3.2 Selmer systems and Selmer groups

For the rest of the chapter, we will study the Galois cohomology groups $H^1(G_K, V)$ for K some number field and $\rho\colon G_K \to V$ a continuous 2-dimensional mod p representation of G_K. We also make the additional restriction that V is absolutely irreducible. The group $H^1(G_K, V)$ is, in general, infinite-dimensional and difficult to work with directly. A Selmer group is a particular subgroup of $H^1(G_K, V)$, chosen in such a way as to be finite, easier to compute with exactly, and ideally carrying some important arithmetic information about the representation V. They are defined using *local conditions*, coming from the completions of K at its various places.

We recall the embeddings $D_{\mathfrak{p}} \hookrightarrow G_K$ from Definition 2.1.11. They give rise to the restriction morphisms of cohomology groups

$$H^1(G_K, V) \to H^1(D_{\mathfrak{p}}, V)$$

for all places $\mathfrak{p}$ of K.

Definition 3.2.1. A **local condition at $\mathfrak{p}$** is a subgroup $L_{\mathfrak{p}} \leq H^1(D_{\mathfrak{p}}, V)$.

There are many local conditions one could define, but we will focus on just a few. The most important local condition is the following.

Definition 3.2.2. The **unramified condition** is the subgroup

$$H^1_{\mathrm{unr}}(D_{\mathfrak{p}}, V) \coloneqq \ker(H^1(D_{\mathfrak{p}}, V) \to H^1(I_{\mathfrak{p}}, V)).$$

This is the subgroup of $H^1(D_{\mathfrak{p}}, V)$ consisting of all cocycle classes that vanish on inertia. Such a cocycle class is called **unramified**. This is the only local condition we need to define a Selmer group.

Definition 3.2.3. A **Selmer system** for V is a set of $\mathcal{L}$ of local conditions $L_{\mathfrak{p}}$, indexed by places $\mathfrak{p}$ of K, such that all but finitely many $L_{\mathfrak{p}}$ are the unramified condition.

Remark 3.2.4. The infinite places ultimately don't find themselves heavily involved in these calculations, although a little work is needed to see why. If $\mathfrak{p}$ is an infinite place, its completion corresponds to either $\mathbb{R}$ or $\mathbb{C}$, which both algebraically close to $\mathbb{C}$, giving possible decomposition groups

$$\mathrm{Gal}(\mathbb{C}/\mathbb{R}) \simeq C_2, \quad \mathrm{Gal}(\mathbb{C}/\mathbb{C}) \simeq C_1.$$

Thus, unless $\mathfrak{p}$ is real and $p = 2$, we will have $H^1(\mathrm{Gal}(\overline{K}_{\mathfrak{P}}/K_{\mathfrak{p}}), V) = 0$, per Remark 3.7 of [Rub00]. For mod 2 Selmer groups, we will allow any amount of ramification at the real places.

Definition 3.2.5. Given a Selmer system $\mathcal{L}$, the **Selmer group attached to** $\mathcal{L}$ is the subgroup of $H^1(G_K, V)$ given by

$$\mathrm{Sel}_{\mathcal{L}}(\rho) \coloneqq \ker\left(H^1(G_K, V) \to \prod_{\mathfrak{p}\nmid\infty} \frac{H^1(D_{\mathfrak{p}}, V)}{L_{\mathfrak{p}}}\right).$$

That is, we keep a class in $H^1(G_K, V)$ if it is in every $L_{\mathfrak{p}}$ after restriction to $D_{\mathfrak{p}}$.

We will see in Section 3.3.3.1 that a Selmer group of a mod p representation is finite. In particular, it is a finite-dimensional vector space over $\mathbb{F}_q$, where $V = \mathbb{F}_q \oplus \mathbb{F}_q$.

Definition 3.2.6. The rank of $\mathrm{Sel}_{\mathcal{L}}$ is its dimension as a vector space over $\mathbb{F}_q$.

We will focus on three Selmer systems. For all three systems we will take $L_{\mathfrak{p}} = H^1_{\mathrm{unr}}(D_{\mathfrak{p}}, V)$ for $\mathfrak{p}$ not over the characteristic p of V, mimicking the definition of the Bloch-Kato Selmer group from Section 3 of [BK07].

3.2.1 The relaxed system

Definition 3.2.7. The **relaxed Selmer system** $\mathcal{L}_{\mathrm{rel}}$ is defined by

$$L_{\mathfrak{p}} = \begin{cases} H^1_{\mathrm{unr}}(D_{\mathfrak{p}}, V) & \text{when } \mathfrak{p} \nmid p \\ H^1(D_{\mathfrak{p}}, V) & \text{when } \mathfrak{p} \mid p. \end{cases}$$

This condition is the most permissive of the three we consider, meaning it defines the largest Selmer group, which we denote $\mathrm{Sel}_{\mathrm{rel}}(\rho)$. This is because we make no condition at places over p: everything in $H^1(G_K, V)$ restricts to 0 in the quotient, because the quotient is trivial.

The relaxed condition is, in general, too permissive to capture "interesting" arithmetic information about ρ; when we transfer Selmer systems to class field theory in Section 3.3, we will see the relaxed system captures extensions of the fixed field of $\ker(\rho)$ that are unramified away from p.

The relaxed Selmer group is still useful, because it is fairly easy to compute. In our algorithms, it exists as a sort of "stepping stone", a group we compute easily that contains a Selmer group defined by finer conditions, whose elements we can examine more closely to "cut down" to our desired group.

3.2.2 The nearly-ordinary system

To define the nearly-ordinary system (see, for example, [EPW05]), we require the following: for each $\mathfrak{p} \mid p$, the restriction of $\rho\colon G_K \to V$ to $D_\mathfrak{p}$ fixes a 1-dimensional subspace $\ell_\mathfrak{p} \leq V$, i.e. a line. Note that these lines $\ell_\mathfrak{p}$ need not be the same subspaces of V for each of the $\mathfrak{p} \mid p$. If such a line exists for a given $\mathfrak{p}$, we say ρ is *nearly-ordinary* at $\mathfrak{p}$.

If ρ is nearly-ordinary, there is a choice of basis of V such that

$$\rho\,|_{D_\mathfrak{p}} \simeq \begin{pmatrix} * & * \\ 0 & * \end{pmatrix}.$$

This also means $\rho^* = \rho\,|_{D_\mathfrak{p}}\colon D_\mathfrak{p} \to V/\ell$ is a character of $D_\mathfrak{p}$.

Definition 3.2.8. The **nearly-ordinary Selmer system** associated to ℓ is

$$L_\mathfrak{p} = \begin{cases} H^1_{\mathrm{unr}}(D_\mathfrak{p}, V) & \text{when } \mathfrak{p} \nmid p \\ \ker(H^1(D_\mathfrak{p}, V) \to H^1(I_\mathfrak{p}, V/\ell)) & \text{when } \mathfrak{p} \mid p. \end{cases}.$$

Since $I_{\mathfrak{p}} \subset D_{\mathfrak{p}}$, the action of $I_{\mathfrak{p}}$ on V/ℓ is well-defined; this system keeps in the Selmer group all those classes that are unramified under the restriction to ρ^*.

Remark 3.2.9. It is possible for $\rho|_{D_{\mathfrak{p}}}$ to fix more than one line in V; in the most extreme case, $D_{\mathfrak{p}}$ is trivial and so $\rho|_{D_{\mathfrak{p}}}$ fixes all $q+1$ lines in V. The Selmer group one obtains from any given fixed line need not be the same as that obtained from any other. In Section 6.3.1 we give an example of a mod 2 representation with three nearly-ordinary Selmer groups, of ranks 1, 2 and 3.

When the specific line used to compute the Selmer group is clear from context, we will write it $\mathrm{Sel}_{\mathrm{NO}}(\rho)$; when there are multiple lines ℓ_i being considered simultaneously, we will write $\mathrm{Sel}_{\mathrm{NO}(\ell_i)}(\rho)$ for the Selmer group attached to ℓ_i.

Remark 3.2.10. The phrase "nearly-ordinary" is used by different authors to mean different things. In some treatments, it is used to describe the slightly stronger condition that $\rho|_{D_{\mathfrak{p}}}$ fixes a line in V **and** is unramified on the quotient V/ℓ. We make no such imposition on the behaviour of $\rho|_{D_{\mathfrak{p}}}$, but we do examine the difference in ranks between the ramified and unramified action on the quotient in Section 6.3.2.

In the case that K has more than one place $\mathfrak{p}$ over p, and that ρ is nearly-ordinary for all of them, we take the nearly-ordinary condition for all of them. Depending on the decomposition groups involved, we can end up with many different Selmer groups, coming from taking different combinations of the various lines fixed by the $\rho\mid_{D_{\mathfrak{p}}}$.

3.2.3 The unramified system

Definition 3.2.11. The **unramified Selmer system** $\mathcal{L}_{\mathrm{unr}}$ takes, for each $\mathfrak{p}$, the local condition $L_{\mathfrak{p}} = H^1_{\mathrm{unr}}(D_{\mathfrak{p}}, V)$.

In contrast to the relaxed condition, the unramified condition is, in general, too restrictive to capture all the "interesting" arithmetic information about ρ. Its elements correspond to everywhere-unramified extensions of the fixed field of $\ker(\rho)$, which are rare.

3.3 Cohomology to class field theory

In this section, we prove an equivalence between lines in $\mathrm{Sel}_{\mathcal{L}}(\rho)$ and extensions of the fixed field of $\ker(\rho)$ satisfying certain properties, coming from the local conditions in $\mathcal{L}$.

3.3.1 Inflation-restriction revisited

Recall the inflation-restriction sequence for a G-module M and $N \triangleleft G$ normal:

$$0 \to H^1(G/N, M^N) \xrightarrow{\text{inf}} H^1(G, M) \xrightarrow{\text{res}} H^1(N, M)^{G/N} \to H^2(G/N, M^N) \to \dots$$

For a mod p representation $\rho\colon G_K \to \mathrm{GL}(V)$, we set $G = G_K$, $N = \ker(\rho)$ and $M = V$. Writing $L = \overline{K}^{\ker(\rho)}$, we note that $G_L = \ker(\rho)$ acts trivially on V, so $H^1(G_L, V) \simeq \mathrm{Hom}(G_L, V)$. Substituting everything in and recalling the isomorphism $\mathrm{Gal}(L/K) \simeq G_K/G_L$ from (2.3), we get the exact sequence

$$0 \to H^1(\mathrm{Gal}(L/K), V) \to H^1(G_K, V) \to \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V) \to H^2(\mathrm{Gal}(L/K), V).$$

The utility of this sequence comes from the fact that it relates cocycles in $H^1(G_K, V)$ to homomorphisms, and that $H^i(\mathrm{Gal}(L/K), V)$ is very often trivial for $i = 1, 2$, see Section 3.4.3. When both are, the middle map

$$H^1(G_K, V) \to \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V) \tag{3.3}$$

is an isomorphism. In particular, when $H^1(\mathrm{Gal}(L/K), V)$ vanishes, the map is injective, and we can bound our Selmer groups from above. Similarly, when $H^2(\mathrm{Gal}(L/K), V)$ vanishes, the map is surjective, and we can bound our Selmer groups below. When both vanish, we can compute the exact rank of our Selmer groups by understanding them as subgroups of $\mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$, which is easier to work with than $H^1(G_K, V)$.

Remark 3.3.1. While $H^i(\mathrm{Gal}(L/K), V)$, $i = 1, 2$, does not vanish for all possible Galois modules V of L/K, it is theoretically possible to work around the fact the non-vanishing prevents an isomorphism. In practice, this will never happen, as we only examine cases where vanishing holds. Thus we may always assume (3.3) is an isomorphism. We have also computed exactly the subgroups for which these cohomology groups do *not* vanish for some small primes, in Section 3.4.3.

3.3.2 Homomorphisms to number fields

Assuming we can move between $H^1(G_K, V)$ and $\mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$ freely, we will now realise elements of the latter as finite extensions M/L satisfying certain properties. In particular, we show the following.

Theorem 3.3.2. Let V be a Galois representation over $\mathbb{F}_p$. Non-trivial lines in $\mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$ are in bijection with extensions M/L satisfying

1. The extension M/L is Galois, with $\mathrm{Gal}(M/L) \simeq V$ (as an additive group);
2. The extension M/K is Galois;
3. The action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(M/L)$ is via ρ.

The main advantage of this theorem is that extensions M/L satisfying points (1-3) are abelian extensions, and once we translate the local conditions, we will see the M/L are unramified for primes of L not over p, meaning they can be found using class field theory, for which extensive and effective algorithms already exist.

Before we prove the theorem, we need to define the action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(M/L)$. Assume that M/K is also Galois, as in the theorem.

Definition 3.3.3. An **extension** of $\sigma \in \mathrm{Gal}(L/K)$ to $\mathrm{Gal}(M/K)$ is any $\tilde{\sigma} \in \mathrm{Gal}(M/K)$ such that $\tilde{\sigma}(x) = \sigma(x)$ for all $x \in L$.

We have an exact sequence of Galois groups

$$1 \to \mathrm{Gal}(M/L) \to \mathrm{Gal}(M/K) \to \mathrm{Gal}(L/K) \to 1,$$

so there are $|\mathrm{Gal}(M/L)|$ choices of extension of $\sigma \in \mathrm{Gal}(L/K)$.

To define how $\sigma \in \mathrm{Gal}(L/K)$ acts on $\tau \in \mathrm{Gal}(M/L)$, choose any extension $\tilde{\sigma}$ of σ to $\mathrm{Gal}(M/K)$, and set

$$(\tau \cdot \sigma)(x) \coloneqq (\tilde{\sigma}^{-1} \circ \tau \circ \tilde{\sigma})(x).$$

We can prove this is an action in the following way: two extensions $\tilde{\sigma}_1, \tilde{\sigma}_2 \in \mathrm{Gal}(M/K)$ agree on L, so $\tilde{\sigma}_1\tilde{\sigma}_2^{-1}$ fixes L, and so is in $\mathrm{Gal}(M/L)$; since $\mathrm{Gal}(M/L)$ is abelian, we are free to rearrange as below to get:

$$\begin{aligned}(\tilde{\sigma}_1^{-1} \circ \tau \circ \tilde{\sigma}_1) \circ (\tilde{\sigma}_2^{-1} \circ \tau \circ \tilde{\sigma}_2)^{-1} &= \tilde{\sigma}_1^{-1} \circ \tau \circ (\tilde{\sigma}_1 \circ \tilde{\sigma}_2^{-1}) \circ \tau^{-1} \circ \tilde{\sigma}_2 \\ &= \tilde{\sigma}_1^{-1} \circ (\tilde{\sigma}_1 \circ \tilde{\sigma}_2^{-1}) \circ \tau \circ \tau^{-1} \circ \tilde{\sigma}_2 \\ &= 1.\end{aligned}$$

Thus the action is independent of the choice of extension.

This is a specific instance of a more general phenomenon: if G is a group with $H \trianglelefteq G$ abelian, then G/H acts on H by conjugation.

Remark 3.3.4. This action can be reinterpreted at the level of the absolute Galois groups G_K, G_L and G_M as follows: take $\sigma \in \mathrm{Gal}(L/K)$ and $\tau \in \mathrm{Gal}(M/L)$. Using the isomorphisms $\mathrm{Gal}(L/K) \simeq G_K/G_L$ (via ρ) and $\mathrm{Gal}(M/L) \simeq G_L/G_M$ (via a homomorphism f in $\mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$), we can lift these to $\tilde{\sigma} \in G_K$ and $\tilde{\tau}$ in G_L. Since $G_L \triangleleft G_K$, the conjugation $\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}$ is still in G_L, and so lies in the domain of f. Since G_K acts on G_L by conjugation, we get

$$\tau \cdot \sigma \coloneqq f(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}).$$

Remark 3.3.5. The condition that V be a representation over $\mathbb{F}_p$, rather than $\mathbb{F}_q$ for some prime power q, is important. Without this condition, the theorem as-stated would not work. We only perform Selmer computations over $\mathbb{F}_p$ in this thesis, so we do not need to worry about $\mathbb{F}_q$ complications.

Proof of Theorem 3.3.2. We start with an element $f \in \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$. This is a homomorphism $G_L \to V$ invariant under the action of $\mathrm{Gal}(L/K) \simeq G_K/G_L$, which acts on f via this identification: take $\tilde{\sigma} \in G_K$ and write σ for its image in the quotient. Then we have

$$(\sigma \cdot f)(\tau) = f(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma})\sigma^{-1}$$

for all $\tilde{\tau} \in G_L$. We note that the action on the right by σ is by ρ. Since f is invariant under this action, we have

$$f(\tilde{\tau}) \cdot \sigma = f(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}). \tag{3.4}$$

We write $M_f = \overline{K}^{\ker(f)}$, and note the following consequences of (3.4):

(I) The group $\mathrm{im}(f)$ is a G_K-stable subspace of V, since $f(\tilde{\tau}) \cdot \sigma = f(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}) \in \mathrm{im}(f)$.

(II) The group $\ker(f)$ is normal not only in G_L, but also in G_K. To show this, choose $\tilde{\tau} \in \ker(f)$. Then $f(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}) = f(\tilde{\tau}) \cdot \sigma = 0 \cdot \sigma = 0$ for all $\sigma \in G_K$.

From point (I) we see that M_f/L is Galois, with Galois group $\mathrm{im}(f) \leq V$. Since V is assumed to be an irreducible G_K-module, and $\mathrm{im}(f) \leq V$ is also a G_K-module, it must either be V or 0. The homomorphism $G_L \to V$ sending everything to 0 corresponds to the trivial extension L/L, so discounting this option (i.e. assuming M_f/L is non-trivial), M_f/L must have Galois group V. If $\alpha \in \mathbb{F}_p$, we have $\ker(\alpha f) = \ker(f)$, so every element in the line αf maps to the same extension $M_f/L = M_{\alpha f}/L$. This gives condition (1) in the theorem.

From point (II) we see that M_f/K is also a Galois extension, giving condition (2) in the theorem.

Finally, to prove condition (3), take $\tau \in \mathrm{Gal}(M/L)$. We have $\mathrm{Gal}(M_f/L) \simeq f(G_L)$, so we can take some $\tilde{\tau} \in G_L$ such that $f(\tilde{\tau}) = \tau$. We want to act on τ by the conjugation action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(M/L)$. Recall from Remark 3.3.4, we can write this as

$$\tau \cdot \sigma = f(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma})$$

for some lift of $\sigma \in \mathrm{Gal}(L/K)$ to G_K. By (3.4), we get that

$$\tau \cdot \sigma = f(\tilde{\tau}) \cdot \sigma$$

where the right-hand-side action is via ρ. But $\tilde{\tau}$ is defined such that $f(\tilde{\tau}) = \tau$, so the left-hand-side action must be via ρ.

For the reverse inclusion, let M/L be an extension satisfying (1-3). To define the required homomorphism, we note that there is an isomorphism $\varphi\colon \mathrm{Gal}(M/L) \to V$, and that $\mathrm{Gal}(M/L) \simeq G_L/G_M$. We claim the required homomorphism is the composition

$$f_M\colon G_L \twoheadrightarrow \mathrm{Gal}(M/L) \xrightarrow{\varphi} V.$$

Certainly this is a homomorphism $G_L \to V$; it is $\mathrm{Gal}(L/K)$-invariant if

$$f_M(\tilde{\tau}) = f_M(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}) \cdot \sigma^{-1}$$

for all $\tilde{\tau} \in G_L$, all $\sigma \in \mathrm{Gal}(L/K)$ with lift $\tilde{\sigma} \in G_K$. In other words, if

$$f_M(\tilde{\tau}) \cdot \sigma = f_M(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}).$$

We have $f_M(\tilde{\tau}) \cdot \sigma = \tau\rho(\sigma)$, since $\sigma \in \mathrm{Gal}(L/K)$ and $\tau = f_M(\tilde{\tau}) \in V$. Meanwhile, we have $\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma} \in G_L$, so $f_M(\tilde{\sigma}^{-1}\tilde{\tau}\tilde{\sigma}) = \sigma'^{-1}\tau\sigma'$, with σ' here denoting a *lift* of $\sigma \in \mathrm{Gal}(L/K)$ to $\mathrm{Gal}(M/K)$. This is exactly the definition of the action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(M/L)$: lifting, then conjugating. Thus, the two actions match, and f_M is $\mathrm{Gal}(L/K)$-invariant. □

3.3.3 *Translating local conditions*

The next step is to turn the local conditions defined in cohomology into properties of the extensions $M_f/L = M/L$ given by Theorem 3.3.2. We tackle the unramified condition first. Recall, for a given place $\mathfrak{p}$ of K, we have an implicitly chosen primes $\mathfrak{q}$ of L and $\mathfrak{P}$ of $\overline{K}$ respectively, such that $\mathfrak{P} \mid_L = \mathfrak{q}$, $\mathfrak{P} \mid_K = \mathfrak{p}$, and $(\mathfrak{P} \mid_L) \mid_K = \mathfrak{p}$.

Proposition 3.3.6. The inertia group $I_{\mathfrak{q}} \leq \mathrm{Gal}(\overline{K}/K)$ can be written

$$I_{\mathfrak{q}} = I_{\mathfrak{p}} \cap G_L.$$

Proof. We write $\mathbb{F}_{\mathfrak{p}}$, $\mathbb{F}_{\mathfrak{q}}$ and $\overline{\mathbb{F}_{\mathfrak{p}}}$ for the residue fields of $K_{\mathfrak{p}}$, $L_{\mathfrak{q}}$ and $\overline{K}_{\mathfrak{P}}$ respectively. Recalling Definition 2.1.13 of the inertia group, we obtain a commutative diagram

$$\begin{tikzcd}
\mathrm{Gal}(\overline{K}_{\mathfrak{P}}/L_{\mathfrak{q}}) \arrow[r, two heads] \arrow[d, hook] & \mathrm{Gal}(\overline{\mathbb{F}_{\mathfrak{p}}}/\mathbb{F}_{\mathfrak{q}}) \arrow[d, hook] \\
\mathrm{Gal}(\overline{K}_{\mathfrak{P}}/K_{\mathfrak{p}}) \arrow[r, two heads] & \mathrm{Gal}(\overline{\mathbb{F}_{\mathfrak{p}}}/\mathbb{F}_{\mathfrak{p}})
\end{tikzcd}$$

with horizontal kernels giving $I_{\mathfrak{q}}$ and $I_{\mathfrak{p}}$. We clearly have $I_{\mathfrak{q}} \leq I_{\mathfrak{p}}$; meanwhile, the commutativity of the diagram (coming from the compatibility of the underlying maximal ideals $\mathfrak{m}_{\mathfrak{p}} \triangleleft \mathcal{O}_{K_{\mathfrak{p}}}$ and $\mathfrak{m}_{\mathfrak{q}} \triangleleft \mathcal{O}_{L_{\mathfrak{q}}}$) gives that $I_{\mathfrak{p}} \cap G_L = I_{\mathfrak{p}} \cap \mathrm{Gal}(\overline{K}_{\mathfrak{P}}/K_{\mathfrak{p}}) \leq I_{\mathfrak{q}}$. $\square$

3.3.3.1 The unramified condition

Here we hope to translate the condition of a cocycle class in $H^1(G_K, V)$ being in $H^1_{\mathrm{unr}}(D_{\mathfrak{p}}, V)$ to a condition on the number field extension M/L. The naive hope is that the extension itself will be unramified at places of L over $\mathfrak{p}$; this is exactly what happens.

Recall from Section 2.1.1, for each place $\mathfrak{p}$ of K and every F/K Galois, we have made a choice of place $\mathfrak{q}$ of F in a compatible way, giving a place $\mathfrak{P}$ of $\overline{K}$ such that $\mathfrak{P}\,|_F = \mathfrak{q}$. We write $\mathfrak{q}$ for the restriction $\mathfrak{P}\,|_L$, and $\mathfrak{q}'$ for the restriction $\mathfrak{P}\,|_M$.

The identification $\mathrm{Gal}(L/K) \simeq G_K/G_L$ lets us write the inertia subgroup $I_{\mathfrak{p}}(L/K)$ as

$$I_{\mathfrak{p}}(L/K) = \mathrm{im}(I_{\mathfrak{p}} \hookrightarrow G_K \overset{\rho}{\twoheadrightarrow} G_K/G_L \simeq \mathrm{Gal}(L/K)) = \rho(I_{\mathfrak{p}}), \tag{3.5}$$

so, in particular, we have

$$I_{\mathfrak{p}}(L/K) \simeq \frac{I_{\mathfrak{p}}}{I_{\mathfrak{p}} \cap G_L} = \frac{I_{\mathfrak{p}}}{I_{\mathfrak{q}}}.$$

Then the restriction map

$$\mathrm{res}\colon H^1(G_K, V) \to H^1(I_{\mathfrak{p}}, V)$$

induces a commutative diagram

$$\begin{tikzcd} H^1(G_K, V) \arrow[r] \arrow[d, "\mathrm{res}"] & \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V) \arrow[d, "\mathrm{res}"] \\ H^1(I_{\mathfrak{p}}, V) \arrow[r] & \mathrm{Hom}_{I_{\mathfrak{p}}(L/K)}(I_{\mathfrak{q}}, V). \end{tikzcd}$$

Let $f \in H^1(G_K, V)$ and denote by $\tilde{f}$ the corresponding element in $\mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$. Assume $f \in H^1_{\mathrm{unr}}(D_{\mathfrak{p}}, V)$, so that $\mathrm{res}(f) = 0$ in $H^1(I_{\mathfrak{p}}, V)$. Then, by the commutativity of the diagram, $\mathrm{res}(\tilde{f}) = 0$. So we have

$$I_q \le \ker(\tilde{f}).$$

Similarly to (3.5), the identification $\mathrm{Gal}(M/L) \simeq G_L/G_M$ allows us to write

$$I_{\mathfrak{q}}(M/L) = \mathrm{im}(I_{\mathfrak{q}} \hookrightarrow G_L \overset{\tilde{f}}{\twoheadrightarrow} G_L/G_M \simeq \mathrm{Gal}(M/L)) = \tilde{f}(I_{\mathfrak{q}}).$$

Thus, $I_{\mathfrak{q}}(M/L) = \tilde{f}(I_{\mathfrak{q}}) = 0$. If $\mathfrak{q}_1$ is a place of L other than $\mathfrak{q} = \mathfrak{P}\,|_L$, it is G_L-conjugate to $\mathfrak{q}$ and so is also in the kernel of $\tilde{f}$. So, in fact, $I_{\mathfrak{q}}(M/L) = 0$ for all places over $\mathfrak{p}$.

Thus we have the following.

Proposition 3.3.7. Let $f \in H^1_{\mathrm{unr}}(D_{\mathfrak{p}}, V)$. Then M_f/L is unramified for all places $\mathfrak{q}$ of L extending $\mathfrak{p}$.

As an immediate consequence, we note the following result.

Theorem 3.3.8. The group $\mathrm{Sel}_{\mathcal{L}}(\rho)$ is finite.

This follows from the classical result that there are only finitely many extensions of a fixed degree with a fixed finite set of ramifying primes. This, in turn, gives us only finitely many possible extensions corresponding to lines in any Selmer group, since one is required to take the unramified condition almost everywhere.

Remark 3.3.9. This argument is technically incomplete if the map (3.3) is not injective. However, since $H^1(\mathrm{Gal}(L/K), V)$ is finite, the kernel of this map is at most finite, so there is no possibility of infinitely-many elements of $\mathrm{Sel}_{\mathcal{L}}(\rho)$ being killed, so the result still follows in this case.

In order to compute the rank of the Selmer group exactly, we require the converse result.

Proposition 3.3.10. Suppose M/L satisfies the conditions of Theorem 3.3.2, and M/L is unramfied at all places $\mathfrak{q}$ extending $\mathfrak{p}$. Then the corresponding homomorphism $\tilde{f} \in \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$ is unramified at $\mathfrak{p}$.

Proof. Recall the homomorphism $\tilde{f}_M \colon G_L \to V$ associated to M is

$$\tilde{f}_M \colon G_L \to G_L/G_M \simeq \mathrm{Gal}(M/L) = V.$$

It is unramified at $\mathfrak{p}$ if $\tilde{f}_M(I_{\mathfrak{p}} \cap G_L) = 0$. Proposition 3.3.6 tells us that $G_L \cap I_{\mathfrak{p}} = I_{\mathfrak{q}}$, and in the proof of Proposition 3.3.7 we showed that $I_{\mathfrak{q}}(M/L) = \tilde{f}_M(I_{\mathfrak{q}})$, so we are done. □

3.3.3.2 The nearly-ordinary condition

We proceed in essentially the same manner as for the unramfied condition in Section 3.3.3.1. Recall that ρ is nearly-ordinary if $\rho\mid_{D_{\mathfrak{p}}}$ fixes a 1-dimensional line $\ell \leq V$.

Equivalently, we have an exact sequence of $D_{\mathfrak{p}}$ modules

$$0 \to \ell \to V \to V/\ell \to 0.$$

The nearly-ordinary condition is then

$$L_{\mathfrak{p}} = \ker(H^1(D_{\mathfrak{p}}, V) \to H^1(I_{\mathfrak{p}}, V/\ell)).$$

This induces a commutative diagram

$$\begin{tikzcd}
H^1(G_K, V) \arrow[r] \arrow[d] & \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V) \arrow[d] \\
H^1(I_{\mathfrak{p}}, V) \arrow[r] \arrow[d] & \mathrm{Hom}_{I_{\mathfrak{p}}(L/K)}(I_{\mathfrak{q}}, V) \arrow[d] \\
H^1(I_{\mathfrak{p}}, V/\ell) \arrow[r] & \mathrm{Hom}_{I_{\mathfrak{p}}(L/K)}(I_{\mathfrak{q}}, V/\ell)
\end{tikzcd}$$

where we are again using that $I_{\mathfrak{q}} = I_{\mathfrak{p}} \cap G_L$. Let $f \in L_{\mathfrak{p}}$, i.e. the image of f in $H^1(I_{\mathfrak{p}}, V/\ell)$ vanishes. Then, by the commutativity of the diagram, the corresponding homomorphism $\tilde{f} \in \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$ vanishes in $\mathrm{Hom}_{I_{\mathfrak{p}}(L/K)}(I_{\mathfrak{q}}, V/\ell)$. Thus, $\tilde{f}(I_{\mathfrak{q}}) \leq \ell$.

As in Section 3.3.3.1, we have, for the place $\mathfrak{q} = \mathfrak{P} |_L$ of L, that

$$I_{\mathfrak{q}}(M/L) = \tilde{f}(I_{\mathfrak{q}}) \leq \ell.$$

So M/L satisfies the nearly-ordinary condition at $\mathfrak{q}$ if $I_{\mathfrak{q}}(M/L) \leq \ell$. Note, the prime $\mathfrak{q}$ needs to be the same prime used to define the decomposition group $D_{\mathfrak{q}}$ that gives rise to the line $\ell \leq V$. If a different $\mathfrak{q}'$ is used, we get a (possibly) different fixed line ℓ' and a (possibly) different inertia group $I_{\mathfrak{q}'}(M/L)$. Due to the Galois action, we have $I_{\mathfrak{q}}(M/L) \leq \ell$ if and only if $I_{\mathfrak{q}'}(M/L) \leq \ell'$.

We again need the converse result to properly compute the rank of the nearly-ordinary Selmer group.

Proposition 3.3.11. Suppose M/L satisfies the conditions of Theorem 3.3.2, and $I_{\mathfrak{q}}(M/L) \leq \ell$. Then the corresponding homomorphism $\tilde{f}_M \in \mathrm{Hom}_{\mathrm{Gal}(L/K)}(G_L, V)$ is trivial in $\mathrm{Hom}_{I_{\mathfrak{p}}(L/K)}(I_{\mathfrak{q}}, V/\ell)$.

Proof. The proof is essentially already done. The homomorphism $\tilde{f}_M$ is

$$\tilde{f}_M \colon G_L \to G_L/G_M \simeq \mathrm{Gal}(M/L) = V.$$

If $\tilde{f}_M(I_{\mathfrak{q}}) = I_{\mathfrak{q}}(M/L) \leq \ell$, then $\tilde{f}_M$ is the zero homomorphism in $\mathrm{Hom}_{I_{\mathfrak{p}}(L/K)}(I_{\mathfrak{q}}, V/\ell)$. □

3.4 Miscellaneous

3.4.1 Comparing Selmer groups

Clearly, $\mathrm{Sel}_{\mathrm{unr}}(\rho), \mathrm{Sel}_{\mathrm{NO}}(\rho) \leq \mathrm{Sel}_{\mathrm{rel}}(\rho)$. In fact, we have

$$\mathrm{Sel}_{\mathrm{unr}}(\rho) \leq \mathrm{Sel}_{\mathrm{NO}}(\rho) \leq \mathrm{Sel}_{\mathrm{rel}}(\rho).$$

This follows from the following observation: the unramified condition is

$$L_{\mathfrak{p}}^{\mathrm{unr}} = \ker(H^1(D_{\mathfrak{p}}, V) \to H^1(I_{\mathfrak{p}}, V)),$$

and the nearly-ordinary condition is

$$L_{\mathfrak{p}}^{\mathrm{NO}} = \ker(H^1(D_{\mathfrak{p}}, V) \to H^1(I_{\mathfrak{p}}, V/\ell)).$$

If ρ is nearly-ordinary at a place $\mathfrak{p}$ over p, then a class $f \in L_{\mathfrak{p}}^{\mathrm{unr}}$ is automatically in $L_{\mathfrak{p}}^{\mathrm{NO}}$, since f being the zero class in $H^1(I_{\mathfrak{p}}, V)$ means f is also the zero class in $H^1(I_{\mathfrak{p}}, V/\ell)$.

3.4.2 Dimension of vector space from number of lines

The following lemma is very easy to prove; we include it for completeness, as we use it extensively in later chapters.

Lemma 3.4.1. Let W be an $\mathbb{F}_q$-vector space of dimension n. The number of lines in W is $\frac{q^n-1}{q-1}$.

3.4.3 Non-vanishing H^1, H^2

Here we note which subgroups of $H \leq \mathrm{GL}_2(\mathbb{F}_p)$ have non-vanishing $H^i(H,V)$, where $V \simeq \mathbb{F}_p \oplus \mathbb{F}_p$ is given an H action by restriction of the standard action of $\mathrm{GL}_2(\mathbb{F}_p)$, for small primes $p = 2, 3, 5$ and $i = 1, 2$. Recall, the non-vanishing of these H^i prevents the map (3.3) from being an isomorphism. So, if a subgroup $H \leq \mathrm{GL}_2(\mathbb{F}_p)$ does not appear here, that map *is* an isomorphism, and the methods of this chapter can be used to exactly compute ranks of Selmer groups attached to representations ρ with $\mathrm{im}(\rho) = H$.

All computations were performed in Magma, using `H1H2_testing.m`, in Section A.2.7.

We list subgroups according to their LMFDB labels, as the isomorphism class of the subgroup is generally not enough to distinguish it as problematic. For more details on the naming conventions for abstract groups and their subgroups in use on the LMFDB, see `https://lmfdb.org/Groups/Abstract/`.

We write h_i for $\dim_{\mathbb{F}_p}(H^i(H,V))$.

3.4.3.1 The case $p = 2$

No subgroup $H \leq \mathrm{GL}_2(\mathbb{F}_2)$ has $H^i(H,V) \neq 0$, for $i = 1, 2$.

3.4.3.2 The case $p = 3$

The group $\mathrm{GL}_2(\mathbb{F}_3)$ has LMFDB label 48.29. It has three subgroups where at least one of H^1 and H^2 does not vanish.

Name	Order	LMFDB	h_1	h_2
C_3	3	16.a1.a1	1	1
S_3	6	8.b1.a2	0	1
S_3	6	8.b1.a1	1	0

3.4.3.3 The case $p = 5$

The group $\mathrm{GL}_2(\mathbb{F}_5)$ has LMFDB label 480.218. It has five subgroups where at least one of H^1 and H^2 does not vanish.

Name	Order	LMFDB	h_1	h_2
C_5	5	96.a1.a1	1	1
D_5	10	48.b1.a1	0	1
D_5	10	48.b1.a2	1	0
F_5	20	24.d1.a1	0	1
F_5	20	24.d1.b2	1	0

3.4.3.4 Primes $p > 5$

For primes $p > 5$, no subgroup data of $\mathrm{GL}_2(\mathbb{F}_p)$ is currently available in the LMFDB. We don't perform any Selmer group computations for these primes, although problematic subgroups can still be found easily using the code referenced above.

4. COHOMOLOGY OF BIANCHI GROUPS

In this chapter, we overview some important results concerning the cohomology of Bianchi groups. Via the Eichler-Shimura isomorphism, these cohomology groups capture important arithmetic information coming from Bianchi modular forms—modular forms over imaginary quadratic fields. We recap methods to compute both H^1 and H^2 of these groups, and define the Hecke action on both.

4.1 Bianchi groups

Let $K = K_d = \mathbb{Q}(\sqrt{-d})$ for $d > 0$ and squarefree, be an imaginary quadratic field with ring of integers $\mathcal{O}_d \coloneqq \mathcal{O}_K$.

Definition 4.1.1. A **Bianchi group** is one of the groups

$$\Gamma_d = \mathrm{PSL}_2(\mathcal{O}_d) = \left\{ \begin{pmatrix} a & b \\ c & e \end{pmatrix} \;\middle|\; a, b, c, e \in \mathcal{O}_d,\ ae - bc = 1 \right\} / \{\pm I\}.$$

When the field K (and so d) is obvious from context, we will also simply write Γ for $\mathrm{PSL}_2(\mathbb{Z}_K)$. We will restrict our attention to the Euclidean Bianchi groups, i.e. those for which $\mathcal{O}_d$ is a Euclidean ring. These are exactly the K_d corresponding to $d \in \{1, 2, 3, 7, 11\}$. We make this restriction for a few reasons, see Remark 4.2.3 and Remark 4.3.1.

We fix the following notation throughout:

$$\omega = \omega_d := \begin{cases} \sqrt{-1}, & d = 1, \\ \sqrt{-2}, & d = 2, \\ \frac{1+\sqrt{-3}}{2}, & d = 3, \\ \frac{1+\sqrt{-7}}{2}, & d = 7, \\ \frac{1+\sqrt{-11}}{2}, & d = 11. \end{cases}$$

Note that $\{1, \omega_d\}$ forms a $\mathbb{Z}$-basis of $\mathcal{O}_d$.

4.1.1 Congruence subgroups of Bianchi groups

The most important subgroups of Bianchi groups (for our purposes) are *congruence subgroups*, which we now define. Let $\mathfrak{n}$ be an ideal of $\mathcal{O}_K$.

Definition 4.1.2. The **principal congruence subgroup of level $\mathfrak{n}$** is the group of matrices

$$\Gamma(\mathfrak{n}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \pmod{\mathfrak{n}} \right\} \leq \mathrm{PSL}_2(\mathcal{O}_K).$$

A **congruence subgroup of level $\mathfrak{n}$** is a subgroup of Γ containing $\Gamma(\mathfrak{n})$. Of particular interest to us is the group $\Gamma_0(\mathfrak{n})$:

$$\Gamma_0(\mathfrak{n}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} * & * \\ 0 & * \end{pmatrix} \pmod{\mathfrak{n}} \right\} \leq \mathrm{PSL}_2(\mathcal{O}_K),$$

We develop the theory of cohomology of Bianchi groups with a general module M, though we are really interested in the groups $H^1(\Gamma_0(\mathfrak{n}), M)$ for some $\Gamma_0(\mathfrak{n})$-module M. The main tool in computing H^1 and H^2 of Γ is its group presentation, and while presentations for $\Gamma_0(\mathfrak{n})$ do exist, they are much more cumbersome to work with.

The alternative to this is to use Shapiro's lemma (Proposition 3.1.8), which we may do since $[\Gamma : \Gamma_0(\mathfrak{n})] < \infty$. Recall, the modified Shapiro lemma gives an isomorphism

$$H^i(\Gamma, R[\Gamma_0(\mathfrak{n})\backslash\Gamma] \otimes M) \simeq H^i(\Gamma_0(\mathfrak{n}), M)$$

for all $\Gamma_0(\mathfrak{n})$-modules M. We now describe the coset space $\Gamma_0(\mathfrak{n})\backslash\Gamma$. This is a standard part of the theory and can be found in many places, see e.g. Section 2.2 of [Cre84].

Definition 4.1.3. For a commutative ring R with identity, the **projective line** over R is the set of ordered pairs $(c, d) \in R^2$, modulo the equivalence relation

$$(a, b) \equiv (c, d) \Leftrightarrow a = uc, b = ud$$

for some $u \in R^\times$. We write $(c : d)$ for the equivalence class of (c, d), and $\mathbb{P}^1(R)$ for the projective line.

Proposition 4.1.4. The right cosets of $\Gamma_0(\mathfrak{n})$ in Γ are in bijection with elements of $\mathbb{P}^1(\mathcal{O}/\mathfrak{n})$. The bijection is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto (c : d).$$

The bijection respects the usual action of Γ on $\Gamma_0(\mathfrak{n})\backslash\Gamma$ (multiplication on the right), and the action on the projective line defined by

$$(c : d)\begin{pmatrix} r & s \\ t & u \end{pmatrix} = (rc + td : sc + ud)$$

The proof can be found in [Cre84].

4.1.2 Modules of Bianchi groups

We recap some important modules relevant to the cohomology of Bianchi groups.

Definition 4.1.5. The **level $\mathfrak{n}$ module** of Γ is the space $R[\Gamma_0(\mathfrak{n})\backslash\Gamma] \simeq R[\mathbb{P}^1(\mathcal{O}/\mathfrak{n})]$. It is so named because it is used to compute with Bianchi modular forms of level higher than 1 when computing Bianchi modular forms in cohomology, per Section 4.1.4.

In the same manner, we introduce the weight module.

Definition 4.1.6. Let $V_k(R)$ denote the space of polynomials in two variables X and Y over a ring R, homogeneous of degree k. The space $V_k(\mathbb{C})$ is a right $\mathrm{GL}_2(\mathbb{C})$-module by the action

$$X^{k-i}Y^i \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix} = (aX+bY)^{k-i}(cX+dY)^i.$$

We will also write V_k when the ring is clear from context. The **(complex) weight** (k,l) **module** is the space

$$V_{k,l}(\mathbb{C}) \coloneqq V_k(\mathbb{C}) \otimes \overline{V_l(\mathbb{C})},$$

where the bar on the second component signifies that the action of $\mathrm{GL}_2(\mathbb{C})$ is twisted by complex conjugation. This module is so named as it allows one to define Bianchi modular forms of higher weight, and compute with them again using the Eichler-Shimura isomorphism (Section 4.1.4).

We can realise this module as complex polynomials in four variables, $X, Y, \overline{X}$ and $\overline{Y}$, homogeneous in each of the pairs X, Y and $\overline{X}, \overline{Y}$, of degrees k and l respectively. Using this representation, we have the action on the second component V_l:

$$\overline{X}^{l-i}\overline{Y}^i \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix} = (\overline{a}\overline{X}+\overline{b}\overline{Y})^{l-i}(\overline{c}\overline{X}+\overline{d}\overline{Y})^i.$$

In Section 5.4.4 we will also use the modules $V_{k,l}(F)$, for number fields F/K. This is a vector space over K, which receives the action of $\mathrm{GL}_2(K)$ as above, identifying the non-trivial Galois automorphism of K with complex conjugation.

Finally, we will also use the modules $V_{k,l}(\mathbb{F}_q)$, where $q = p^r$ for some prime p. In order to define the twisting on the second component, these modules, we start with an ideal $\mathfrak{p} \mid p\mathcal{O}_K$, (or $p\mathcal{O}_F$ for some extension F/K), act on $V_{k,l}(\mathcal{O}_K)$, then reduce modulo $\mathfrak{p}$.

The space $V_{k,l}(\mathbb{C})$ comes equipped with a natural pairing, arising from the pairings on its component spaces. On V_k we have the pairing given by

$$(X^{k-\alpha}Y^{\alpha}, X^{k-\beta}Y^{\beta}) \mapsto (-1)^{\alpha}\binom{k}{\alpha}^{-1}\delta_{\alpha+\beta,k}.$$

The space $\overline{V_l}$ has the analogous pairing, and their product gives the pairing $\langle\cdot,\cdot\rangle$ on $V_{k,l}$

$$\langle X^{k-\alpha}Y^{\alpha}\overline{X}^{l-\gamma}\overline{Y}^{\gamma}, X^{k-\beta}Y^{\beta}\overline{X}^{l-\varepsilon}\overline{Y}^{\varepsilon}\rangle = (-1)^{\alpha+\gamma}\binom{k}{\alpha}^{-1}\binom{l}{\gamma}^{-1}\delta_{\alpha+\beta,k}\delta_{\gamma+\varepsilon,l}.$$

It can alternatively be described by the formula

$$\left\langle (aX+bY)^k(e\overline{X}+f\overline{Y})^l, (cX+dY)^k(g\overline{X}+h\overline{Y})^l\right\rangle = (ad-bc)^k(eh-fg)^l, \quad (4.1)$$

from which follows the relation

$$\langle P\cdot g, Q\rangle = \langle P, Q\cdot g^{\iota}\rangle, \quad (4.2)$$

where $g \in \mathrm{GL}_2(\mathbb{C})$ and $g^{\iota} = \det(g)g^{-1}$. In particular, the dual of $g \in \mathrm{SL}_2(\mathbb{C})$ is g^{-1}, so the pairing is Γ_d-invariant for all our d.

Remark 4.1.7. The same pairings can be defined on $V_{k,l}(\mathbb{F}_{\mathfrak{p}})$ for some prime $\mathfrak{p}$ of $\mathcal{O}_K$, however one must make the restriction that $p \nmid k, l$, so that the pairing is actually defined. In general, one is only interested in the modules $V_{k,l}(\mathbb{F}_{\mathfrak{p}})$ when $0 \leq k, l < p$, as beyond this range they are no longer irreducible.

We will also use several twists of the weight module. The first kind of twists are the **determinant twists**: for two integers s, t, we twist the action of γ on $V_{k,l}$ by the determinant of γ:

$$X^{k-i}Y^{i}\overline{X}^{l-i}\overline{Y}^{i}\cdot\gamma = \det(\gamma)^s(X^{k-i}Y^{i}\cdot\gamma)\overline{\det(\gamma)}^t(\overline{X}^{l-i}\overline{Y}^{i}\cdot\gamma). \quad (4.3)$$

We write the twisted module $V_{k,l}^{s,t}$. Note that $V_{k,l}^{0,0} = V_{k,l}$.

Finally, we will also need to incorporate the action of a Dirichlet character of $(\mathcal{O}_K/\mathfrak{n})^\times$. For $\gamma = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \Gamma_0(\mathfrak{n})$, we define the action of γ on $V_{k,l}^{s,t,\chi}(\mathbb{C})$ by the action (4.3) of γ on $V_{k,l}^{s,t}(\mathbb{C})$, twisted by the value $\chi(\gamma) \coloneqq \chi(d)$:

$$X^{k-i}Y^i\overline{X}^{l-i}\overline{Y}^i \cdot \gamma = \chi(\gamma)\det(\gamma)^s(X^{k-i}Y^i \cdot \gamma)\overline{\det(\gamma)}^t(\overline{X}^{l-i}\overline{Y}^i \cdot \gamma).$$

Any Dirichlet character $\chi\colon (\mathcal{O}_K/\mathfrak{n})^\times \to \mathbb{C}$ can be realised as a *mod p character* $\overline{\chi}\colon (\mathcal{O}_K/\mathfrak{n})^\times \to \mathbb{F}_q^\times$ by realising its values in the ring of integers of a cyclotomic field F, then reducing modulo an ideal $\mathfrak{p} \mid p$. In this way, we can define the weight module $V_{k,l}^{s,t,\chi}$ modulo p.

4.1.3 Hyperbolic 3-space

Write $\mathcal{H}_3$ for the model of hyperbolic 3-space given by

$$\mathcal{H}_3 = \{(z,t) \mid z \in \mathbb{C}, t \in \mathbb{R}_{>0}\}.$$

The Bianchi groups Γ_d inherit an action on $\mathcal{H}_3$ from a transitive isometric action of $\mathrm{SL}_2(\mathbb{C})$. To define this action, we can realise $\mathcal{H}_3$ as a subspace of the Hamiltonian quaternions

$$\mathbb{H} = \left(\frac{-1,-1}{\mathbb{R}}\right) = \{a + bi + cj + dk \mid a,b,c,d \in \mathbb{R}, i^2 = j^2 = k^2 = ijk = -1\}$$

by sending (z,t) to $\tau = z + tj$. Then the action is via fractional linear transformations

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \tau = (a\tau + b)(c\tau + d)^{-1}.$$

Our primary use of this action is in Section 4.3.2.

4.1.4 The Eichler-Shimura isomorphism

For the most part, we will not use Bianchi modular forms considered as harmonic differential forms on $\mathbb{H}_3$, instead simply considering cohomology classes. We do this using the Eichler-Shimura isomorphism, proved for Bianchi modular forms by Harder [Har75], which we note here for the sake of completeness.

Theorem 4.1.8 (Generalised Eichler-Shimura isomorphism)**.** Write $S_{k+2}(\Gamma')$ for the weight $(k+2, k+2)$ Bianchi modular forms $(k \geq 0)$ for $\Gamma' \leq \Gamma_d = \mathrm{PSL}_2(\mathcal{O}_d)$ a congruence subgroup. Then

$$S_{k+2}(\Gamma') \simeq H^1_{\mathrm{par}}(\Gamma', V_{k,k}(\mathbb{C})) \simeq H^2_{\mathrm{par}}(\Gamma', V_{k,k}(\mathbb{C}))$$

as Hecke modules. Here the subscript par indicates we are working with the *parabolic* cohomology, see Section 4.2.2.

This justifies our treatment of classes in $H^i(\Gamma', V_{k,k}(\mathbb{C}))$ (or with coefficients in $V_{k,k}(\mathcal{O}_d)$) as essentially equivalent to Bianchi modular forms. See Sections 4.2, 4.3 for the definitions of H^1 and H^2.

4.1.5 Presentations of Bianchi groups

For the five d giving Euclidean Bianchi groups, we record a presentation of Γ_d. In all cases, we will make the identifications of generators

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad T_\omega = \begin{pmatrix} 1 & \omega_d \\ 0 & 1 \end{pmatrix}.$$

The matrix S has order 2 in $\mathrm{PSL}_2(\mathcal{O}_K)$, and T, T_ω have infinite order. Additionally, in the cases $d = 1, 3$, the unit group of $\mathcal{O}_K$ is larger than $\{\pm 1\}$. Writing ω_d for the generator of $\mathcal{O}_K^\times$, the presentations for $d = 1, 3$ will include the additional generator

$$L = \begin{pmatrix} \omega_d & 0 \\ 0 & \omega_d^{-1} \end{pmatrix}.$$

This matrix has order 2 when $d = 1$ and order 3 when $d = 3$. All presentations come from [Fin89] unless otherwise specified.

4.1.5.1 The case $d = 1$

$$\Gamma_1 \simeq \langle S, T, T_\omega, L \mid S^2 = L^2 = (SL)^2 = (TL)^2 = (T_\omega L)^2 = (TS)^3 = (T_\omega SL)^3 = [T, T_\omega] = 1\rangle.$$

Remark 4.1.9. Once in matrix form it can be easily verified that L is not strictly necessary for the presentation, as it can be written $ST_\omega^{-1}ST_\omega ST_\omega^{-1}$. However, this substitution makes the presentation more complicated, so we leave it as-is.

4.1.5.2 The case $d = 2$

$$\Gamma_2 \simeq \langle S, T, T_\omega \mid S^2 = (TS)^3 = [T, T_\omega] = [S, T_\omega]^2 = 1\rangle.$$

4.1.5.3 The case $d = 3$

$$\Gamma_3 \simeq \langle S, T, T_\omega, L \mid S^2 = (TS)^3 = (SL)^2 = (T^{-1}T_\omega SL)^3 = L^3 = L^{-1}T^{-1}T_\omega LT^{-1} = L^{-1}TLT_\omega = [T, T_\omega] = 1\rangle$$

We note that this presentation is slightly different than the one appearing in [Fin89]. This difference comes from the different basis for $\mathcal{O}_K$ used, and amounts to replacing occurrences of U in their presentation with $T^{-1}T_\omega$. Our choice of basis of $\mathcal{O}_K$ comes from matching conventions in use on the LMFDB [LMF23].

Remark 4.1.10. As in the case $d = 1$, the generator L is not strictly necessary, as it can be written $STST_\omega ST_\omega^{-1}ST^{-1}$, but for the same reason we leave it as-is.

4.1.5.4 The case $d = 7$

$$\Gamma_7 \simeq \langle S, T, T_\omega \mid S^2 = (TS)^3 = [T, T_\omega] = (T_\omega^{-1} S T_\omega S T)^2 = 1 \rangle.$$

4.1.5.5 The case $d = 11$

$$\Gamma_{11} \simeq \langle S, T, T_\omega \mid S^2 = (TS)^3 = [T, T_\omega] = (T_\omega^{-1} S T_\omega S T)^3 = 1 \rangle.$$

4.2 First cohomology—H^1

The first cohomology of any finitely-presented group G with coefficients in a finite-dimensional G-module M is explicitly computable in finite time. This is done using **Fox calculus**, to turn relations in the presentation into conditions on the cocycles. We illustrate the method on $\Gamma = \Gamma_2$; the general treatment is given in detail in, e.g. Section 8.2 of [Ş08].

Recall from Section 3.1, first cohomology $H^1(G, M)$ is the quotient of the space of 1-cocycles

$$Z^1(G, M) = \{f \colon G \to M \mid f(gh) = f(g) \cdot h + f(h) \text{ for all } g, h \in G\}$$

by the space of 1-coboundaries

$$B^1(G, M) = \{f_m \colon G \to M \mid f_m \colon g \mapsto m \cdot g - m \text{ for some } m \in M, \text{ all } g \in G\}.$$

Writing 1_G for the identity of G, the 1-cocycle condition implies

$$f(1_G) = f(1_G^2) = f(1_G) \cdot 1_G + f(1_G) = 2f(1_G),$$

so $f(1_G) = 0$ for all 1-cocycles f. This process of applying the cocycle relation is the essence of the Fox calculus.

Example 4.2.1. Let $K = \mathbb{Q}(\sqrt{-2})$, with $\mathcal{O}_K = \mathbb{Z}[\omega]$, where $\omega = \sqrt{-2}$, with associated Bianchi group

$$\Gamma_2 \simeq \langle S, T, T_\omega \mid S^2 = (TS)^3 = [T, T_\omega] = [S, T_\omega]^2 = 1 \rangle.$$

Using relations from the group, we can find conditions on the values of a cocycle $f : \Gamma_2 \to M$. For example, the relationship $S^2 = 1$ translates to

$$0 = f(S^2) = f(S)S + f(S) = f(S)[1 + S].$$

So whichever $m \in M$ is mapped to by S under f, it must satisfy $m = -mS$. Similarly, we can translate the other relations of the Bianchi group into conditions on cocycles:

- $f(S)[1 + S] = 0$
- $f(T)[S((TS)^2 + TS + 1)] + f(S)[(TS)^2 + TS + 1] = 0$
- $f(T)[T_\omega T^{-1}T_\omega^{-1} - T^{-1}T_\omega^{-1}] + f(T_\omega)[T^{-1}T_\omega^{-1} - T_\omega^{-1}] = 0$
- $f(S)[T_\omega S T_\omega^{-1} S T_\omega S T_\omega^{-1} + T_\omega^{-1} S T_\omega S T_\omega^{-1} + T_\omega S T_\omega^{-1} + T_\omega^{-1}]$
  $+f(T_\omega)[S T_\omega^{-1} S T_\omega S T_\omega^{-1} - T_\omega^{-1} S T_\omega S T_\omega^{-1} + S T_\omega^{-1} - T_\omega^{-1}] = 0$

Here we have made use of the identity

$$f(g^{-1}) = -f(g)g^{-1}$$

which, again, follows from the cocycle relation. In practice, these conditions can be simplified using the relations defining Γ_2.

These conditions exactly define the space of cocycles on Γ_2: if f is any function that happens to satisfy these relations, it *must* be a cocycle, since any element in the group is a word in the generators S, T and T_ω; conversely, if f is a cocycle, it must be in this space since it satisfies the cocycle relationship for every word in the group, so in particular, it satisfies the cocycle relationship for the words giving each of the four relations used here.

Since M is finite-dimensional, its elements can be represented by vectors with coefficients in its base ring R, so each of the values $f(S)$, $f(T)$, $f(T_\omega)$ can be viewed as vectors over R. By computing the action of Γ_2 on M explicitly with respect to this basis, we can find a block vector $(f(S) \mid f(T) \mid f(T_\omega))$ that is in the kernel of the block matrix

$$\left(\begin{array}{c|c|c|c} r_{1,T} & r_{2,T} & r_{3,T} & r_{4,T} \\ \hline r_{1,S} & r_{2,S} & r_{3,S} & r_{4,S} \\ \hline r_{1,T_\omega} & r_{2,T_\omega} & r_{3,T_\omega} & r_{4,T_\omega} \end{array}\right)$$

where we have rewritten the cocycle condition coming from the i^{th} relation as $f(T)r_{i,T} + f(S)r_{i,S} + f(T_\omega)r_{i,T_\omega} = 0$ (so that $r_{1,S} = 1 + S$, $r_{2,T_\omega} = T^{-1}T_\omega^{-1} - T_\omega^{-1}$, etc.)

This matrix is explicitly computable, and its kernel is isomorphic to the space $Z^1(\Gamma_2, M)$. To compute the 1-coboundaries, we use a similar trick. For every $m \in M$ we get a coboundary of the form $g \mapsto mg - m$, and every coboundary is of this form. So every element of the space $B^1(\Gamma_2, M)$ is captured by a vector lying in the image of the block matrix

$$(1 - T \mid 1 - S \mid 1 - T_\omega).$$

This can also be explicitly computed, and the quotient of the two spaces gives $H^1(\Gamma_2, M)$.

4.2.1 Hecke action on H^1

We now describe how to compute the Hecke action on the space $H^1(\Gamma, M)$, specialising the general account of [AS86]. Let $\mathfrak{q}$ be a prime ideal of $\mathcal{O}_K$, generated by some element $\pi \in \mathcal{O}_K$. Here we assume that, if M has any level structure coming from $\Gamma_0(\mathfrak{n})$, we have $\gcd(\mathfrak{q}, \mathfrak{n}) = 1$.

We write $g = \left(\begin{smallmatrix} \pi & 0 \\ 0 & 1 \end{smallmatrix}\right)$, and note that $\Gamma_0(\mathfrak{q}) = g^{-1}\Gamma g \cap \Gamma$, and, writing $\Gamma^0(\mathfrak{n})$ for the subgroup of matrices with top-right entry $\equiv 0 \pmod{\mathfrak{n}}$, we have $\Gamma^0(\mathfrak{q}) = g\Gamma g^{-1} \cap \Gamma$.

The two groups are related by conjugation:

$$\Gamma_0(\mathfrak{q}) = g^{-1}\Gamma^0(\mathfrak{q})g.$$

The double coset $\Gamma g \Gamma$ decomposes as a disjoint union over representatives gr_i, where r_i are a set of right coset representatives of $\Gamma_0(\mathfrak{q})$ in Γ:

$$\Gamma g \Gamma = \coprod_{i=1}^{k} \Gamma g r_i.$$

The Hecke operator $T_\mathfrak{q}$ is defined as the composition of the maps res, conj and cores as in the diagram

$$\begin{tikzcd}
H^1(\mathrm{PSL}_2(\mathcal{O}_K), M) \arrow[r, dashed, "T_\mathfrak{q}"] \arrow[d, "\mathrm{res}"] & H^1(\mathrm{PSL}_2(\mathcal{O}_K), M) \\
H^1(\Gamma_0(\mathfrak{q}), M) \arrow[r, "\mathrm{conj}"] & H^1(\Gamma^0(\mathfrak{q}), M) \arrow[u, "\mathrm{cores}"']
\end{tikzcd}$$

The maps are as follows:

- res is the map induced by the restriction map $\mathrm{PSL}_2(\mathcal{O}_K) \to \Gamma_0(\mathfrak{q})$.
- conj is the map induced by the conjugation map $\gamma \mapsto g\gamma g^{-1}$.
- cores is the map sending $f \in H^1(\Gamma^0(\mathfrak{q}), M)$ to the cocycle

$$\mathrm{cores}(f) : \gamma \mapsto \sum_{i=1}^{k} f(r_i \gamma r_j^{-1}) r_j,$$

where r_j is the coset representative such that $\Gamma_0(\mathfrak{q}) r_i \gamma = \Gamma_0(\mathfrak{q}) r_j$.

Putting all this together, we have

$$f \cdot T_q : \gamma \mapsto \sum_{i=1}^{k} f(g r_i \gamma r_j^{-1} g^{-1}) g r_j.$$

The set of all Hecke operators for primes $\mathfrak{q}$ coprime to the level of M are known to commute, and in fact can be simultaneously diagonalised. We say $f \in H^1(\Gamma, M)$ is a **Hecke eigenclass** if it is a simultaneous eigenvector for all $T_\mathfrak{q}$ for $\mathfrak{q}$ coprime to the level.

Remark 4.2.2. In fact, the above construction is carried out on cocycle *classes*. The Hecke action stabilises the space $B^1(\Gamma, M)$, so one computes it in practice by starting with a class $[f] \in H^1(\Gamma, M)$, lifting it to a representative cocycle f, computing $f \cdot T_\mathfrak{p}$, and writing the class $[f \cdot T_\mathfrak{p}] \in H^1(\Gamma, M)$ in terms of a chosen basis of $H^1(\Gamma, M)$.

Remark 4.2.3. Since we compute the cocycle f as its image on only the generators of Γ, evaluating $f(M)$ for arbitrary $M \in \Gamma$ requires a word decomposition algorithm. This is one of the places we use the Euclidean property of Γ; the Euclidean algorithm on $\mathcal{O}_K$ can be extended to a word decomposition algorithm, see Section 8.2.4 of [Ş08].

4.2.2 Parabolic cohomology

Definition 4.2.4. The **parabolic subgroup** of a Bianchi group Γ is the group of upper-triangular elements:

$$\Gamma_\infty = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} * & * \\ 0 & * \end{pmatrix} \right\} \leq \Gamma.$$

This group gets its name from the action of Γ on the hyperbolic 3-space. Under this action, the parabolic elements fix the point at infinity.

Definition 4.2.5. A **parabolic 1-cocycle** is an $f \in Z^1(\Gamma, M)$ such that $f(g) = 0$ for all $g \in \Gamma_\infty$. Similarly, a **parabolic 1-coboundary** is an $f \in B^1(\Gamma, M)$ vanishing on all parabolic elements. We write the spaces of parabolic 1-cocycles and 1-coboundaries as $Z^1_{\text{par}}(\Gamma, M)$ and $B^1_{\text{par}}(\Gamma, M)$ respectively.

It is immediate consequence that

$$B^1_{\text{par}}(\Gamma, M) \leq Z^1_{\text{par}}(\Gamma, M)$$

Definition 4.2.6. The **parabolic cohomology of G with coefficients in M** is the space

$$H^1_{\mathrm{par}}(\Gamma, M) = \frac{Z^1_{\mathrm{par}}(\Gamma, M)}{B^1_{\mathrm{par}}(\Gamma, M)}.$$

We will use parabolic cohomology in Section 5.1 to define the space of period polynomials over K.

Remark 4.2.7. To define H^2_{par}, one defines parabolic cocycle and coboundaries in the analogous way, via vanishing on Γ_∞. Our treatment of second cohomology in the next section will not proceed via cocycles and coboundaries, however, so we will not go into further detail here.

4.3 Second cohomology—H^2

Here we introduce the second cohomology of Bianchi groups as quotients of their coefficient modules, relate these groups to modular symbols, and introduce the Hecke action via Heilbronn matrices. In Chapter 5, this will be used to derive a duality between H^2 and period polynomials, giving a Hecke action there.

4.3.1 Second cohomology groups as quotients

We will not use the description of $H^2(G, M)$ in terms of 2-cocycles and 2-coboundaries from Section 3.1, as it does not lend itself well to explicit computation. Instead, specialising to our groups Γ_d, we will use expressions for $H^2(\Gamma, M)$ as particular quotients of M. These come from spectral sequences arising from geometric data relating to the action of Γ_d on the hyperbolic 3-space $\mathcal{H}_3$. For details, see [SV83] and [Ş11].

4.3.1.1 The case $d = 1$

Let $E = T_\omega SL = \left(\begin{smallmatrix} -1 & \omega \\ \omega & 0 \end{smallmatrix}\right)$, which has order 3. For any Γ_1-module M, we have

$$H^2(\Gamma_1, M) \simeq M/(M^S + M^{SL} + M^U + M^E). \tag{4.4}$$

4.3.1.2 The case $d = 2$

Let $A = T_\omega^{-1} S T_\omega S = \left(\begin{smallmatrix} 1 & \omega \\ \omega & -1 \end{smallmatrix}\right)$, which has order 2, and satisfies $A = T_\omega \overline{A} T_\omega^{-1}$. For any Γ_2-module M, we have

$$H^2(\Gamma_2, M) \simeq M/(M^U + M^S + M^A(1 - T_\omega^{-1})) \tag{4.5}$$

4.3.1.3 The case $d = 3$

In Γ_3, both SL and LS have order 2. For any Γ_3-module M, we have

$$H^2(\Gamma_3, M) \simeq M/(M^{LS} + M^U + M^{SL}).$$

4.3.1.4 The case $d = 7$

Let $A = ST_\omega^{-1} S T_\omega T^{-1}$, which has order 2, and $g = ST_\omega^{-1}$, which satisfies $g^{-1}Ag = \overline{A}$. For any Γ_7-module M, we have

$$H^2(\Gamma_7, M) \simeq M/(M^S + M^U + M^A(1 + g)).$$

4.3.1.5 The case $d = 11$

Let $A = ST_\omega^{-1}ST_\omega T^{-1}$, which has order 3, and $g = ST_\omega^{-1}$, which satisfies $g^{-1}Ag = \overline{A}$. For any Γ_{11}-module M, we have

$$H^2(\Gamma_{11}, M) \simeq M/(M^S + M^U + M^A(1+g)).$$

4.3.2 Modular symbols

We define the space $\mathcal{M}_2$ of **weight** 2 **modular symbols** to be the quotient of the complex vector space with basis given by the symbols $\{\alpha, \beta\}$, for $\alpha, \beta \in \mathbb{P}^1(K) = K \cup \{\infty\}$, by the subspace spanned by the elements of the form $\{\alpha,\beta\} + \{\beta,\alpha\}$ and $\{\alpha,\beta\} + \{\beta,\gamma\} + \{\gamma,\alpha\}$. Put more compactly,

$$\mathcal{M}_2 := \mathbb{C}[\{\alpha,\beta\}] \Big/ \Big\langle \{\alpha,\beta\} + \{\beta,\alpha\},\ \{\alpha,\beta\} + \{\beta,\gamma\} + \{\gamma,\alpha\} \Big\rangle.$$

The group $\mathrm{PSL}_2(K)$ acts naturally on $\mathcal{M}_2$ on the right via the formula

$$\{\alpha,\beta\} \cdot g = \{g^{-1}\alpha, g^{-1}\beta\}$$

where $g^{-1}\alpha$ and $g^{-1}\beta$ are the linear fractional transformations of Section 4.1.3.

Theorem 2.6 of [Moh14] tells us[1] that the linear map

$$\Phi^{k,l} = \Phi^{k,l}_d \colon V_{k,l} \longrightarrow (\mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l})_{\Gamma_d}\,, \tag{4.6}$$

given by

$$\Phi^{k,l}(v) = [\{0,\infty\} \otimes v]$$

is a *surjection*. Here the right hand side is the space of coinvariants

$$(\mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l})_{\Gamma_d} = \mathcal{M}_2 \otimes_{\mathbb{C}[\Gamma_d]} V_{k,l} = (\mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l})/\langle u - ua \mid a \in \mathbb{C}[\Gamma_d]\rangle,$$

[1] In our case, his $r = 1$ and his σ_1 equals our matrix S.

with Γ_d and $\mathbb{C}[\Gamma_d]$ acting diagonally on $\mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l}$. Moreover, $[\{0,\infty\} \otimes v]$ denotes the class of $\{0,\infty\} \otimes v$. For convenience, we will call the above map $\Phi_d^{k,l}$ the **Manin surjection** of weight (k,l) for Γ_d. Note that this is not standard terminology.

Remark 4.3.1. It is here also that we use the restriction to the Euclidean fields. The Manin surjection is defined using all the non-parabolic generators of the associated Bianchi group. For a non-Euclidean imaginary quadratic field, there are more non-parabolic generators than just S. For example, for $K = \mathbb{Q}(\sqrt{-19})$, there are two non-parabolic generators, and so the Manin surjection has $V_{k,l} \oplus V_{k,l}$ as its domain, per Sections 2.1, 2.2 of [Moh14]. In principle, it should be possible to extend the methods we use to the non-Euclidean cases, however we leave this as a possibility for future work.

4.3.3 Borel-Serre duality

The (complex) *Steinberg module* St of an arithmetic group G is the $\mathbb{C}[G]$-module $H^{\mathrm{vcd}}(G, \mathbb{C}[G])$ where vcd is the virtual cohomological dimension of G. The Steinberg module is the dualising module of G in the sense that it implements the duality theorem of Borel-Serre (Theorem 11.4.3 of [BS73]): for any $\mathbb{C}[G]$-module M, we have an isomorphism

$$H_j(G, \mathrm{St} \otimes_{\mathbb{C}} M) \simeq H^{\mathrm{vcd}-j}(G, M).$$

This isomorphism respects the action of Hecke operators.

For Bianchi groups Γ, the virtual cohomological dimension is 2, and we get the following corollary: for all k, l

$$H_0(\Gamma, \mathrm{St} \otimes_{\mathbb{C}} V_{k,l}) \simeq H^2(\Gamma, V_{k,l}). \tag{4.7}$$

We will use this fact below.

The Steinberg module is well-known to be related to modular symbols; in our case of

Bianchi groups, the Steinberg module is isomorphic to the space of weight 2 modular symbols that we defined above (see e.g. [Ash94], Definition 4 of [Tor12]). We record this below.

Lemma 4.3.2. For our Bianchi groups Γ, we have an isomorphism $\mathrm{St} \simeq \mathcal{M}_2$ of Γ-modules.

The details of this proof can be found in [Moh14], Section 4.1.1, although the definition of the Steinberg module is different to what we use here; the equivalence of the two definitions can be found in Section 1 of [Ash94].

4.3.4 The kernel of $\Phi_d^{k,l}$

We will now argue that the kernel of the Manin surjection is exactly the subspace that was quotiented out in the previous explicit description of the second cohomology group in Section 4.3.1. See also Proposition 1 of [Tor12].

We begin with a lemma.

Lemma 4.3.3. Let $g \in \Gamma_d$ be an element of finite order, say n. Let $v \in V_{k,l}$ be fixed by g. Then v lies in the kernel of $\Phi_d^{k,l}$.

Proof. Let us put $A = 1 - g$ and $B = (1 + g + \ldots + g^{n-1})$ for convenience. Observe that since $BA = 0$ in the group algebra $\mathbb{C}[\Gamma_d]$, the kernel of A, as a linear map on $V_{k,l}$ equals the image of B. As v is fixed by g, it lies in the kernel of A and hence there is v' such that $v = v'B$. We have

$$\Phi(v) = [\{0,\infty\} \otimes v] = [\{0,\infty\} \otimes v'B] = [\{0,\infty\} \cdot A \otimes v'BA] = [0]$$

as claimed. Note, the penultimate equality follows from the fact that the image of $\Phi_d^{k,l}$ lies in the space $(\mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l})_{\Gamma_d}$ of coinvariants, and so $x = x \cdot A$ (with the diagonal action) for all x. $\square$

4.3.4.1 The cases $d = 1, 3$

First, we treat $d = 1$, by showing that

$$\ker(\Phi_1^{k,l}) = V_{k,l}^S + V_{k,l}^{SL} + V_{k,l}^U + V_{k,l}^E, \tag{4.8}$$

where the right hand side is as in the right hand side of (4.4).

Since the elements S, SL, U, and E are all of finite order, Lemma 4.3.3 implies that the right hand side of (4.8) is a subspace of the left hand side. To show equality, we will argue that they have the same dimension. We first employ Lemma 4.3.2 to view Φ_k^1 as a surjection onto

$$H_0(\Gamma_1, \mathrm{St} \otimes_{\mathbb{C}} V_{k,l}) \simeq H_0(\Gamma_1, \mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l}) \simeq (\mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l})_{\Gamma_1}.$$

Now, it follows from the isomorphisms (4.7) and (4.4) that the dimension of $\ker(\Phi_1^{k,l})$ has to be equal to that of its subspace $V_{k,l}^S + V_{k,l}^{SL} + V_{k,l}^U + V_{k,l}^E$. For $d = 3$, exactly the same logic applied to LS, U and SL gives the result.

4.3.4.2 The cases $d = 2, 7, 11$

These cases are essentially the same, only also relying on the following lemma.
Lemma 4.3.4. *Let $A, B, g \in \Gamma$ such that $g^{-1}Bg = A$. Then $V^A = V^B g$.*

Proof. First we show $V^A \subseteq V^B g$. Take $v \in V^A$. Then $v = v \cdot A = v \cdot g^{-1}Bg$. Multiplying by g^{-1} on both sides, we get $v \cdot g^{-1}B = v \cdot g^{-1}$, i.e. $v \cdot g^{-1} \in V^B$, so $v \in V^B g$. The reverse inclusion follows the same argument, interchanging A with B and g with g^{-1}. □

For $d = 2$, it is noted in Section 4.5 that $A = T_\omega \overline{A} T_\omega^{-1}$, and a quick check shows $A^2 = \overline{A}^2 = 1$, so $V_{k,l}^A(1 - T_\omega^{-1}) = V_{k,l}^A + V_{k,l}^{\overline{A}}$ by Lemma 4.3.4, and so the same argument as above shows

$$\ker(\Phi_2^{k,l}) = V_{k,l}^S + V_{k,l}^U + V_{k,l}^A(1 - T_\omega^{-1}).$$

For $d = 7, 11$ the same argument works, again noting the relations between A and $\overline{A}$ given in Sections 4.3.1.4, 4.3.1.5, giving

$$\ker(\Phi_7^{k,l}) = V_{k,l}^S + V_{k,l}^U + V_{k,l}^A(1 + ST_\omega^{-1})$$

and

$$\ker(\Phi_{11}^{k,l}) = V_{k,l}^S + V_{k,l}^U + V_{k,l}^A(1 + ST_\omega^{-1}).$$

4.3.5 Hecke action on H^2

The construction for Hecke operators acting on cohomology classes in H^2 analogous to that in Section 4.2.1 works, but 2-cocycles and 2-coboundaries are generally much less convenient to work with, so we instead compute the Hecke action on H^2 using **Heilbronn matrices**.

Following Section 3.2 of [Moh14] and Chapter 2 of [Cre97], for a given prime element[2] $\pi \in \mathcal{O}_K$, we define the associated set of **Heilbronn matrices** as

$$H_\pi = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in M_2(\mathcal{O}) \;\middle|\; N(a) > N(b) \geq 0, N(d) > N(c) \geq 0, ad - bc = \pi \right\}$$

Then, for $v \in V_{k,l}$, we define the operator $\widetilde{T}_\pi$ as

$$v \cdot \widetilde{T}_\pi := \sum_{g \in H_\pi} v \cdot g.$$

It is proven by Mohamed in [Moh14] that for any prime element $\pi \in \mathcal{O}$, we have a

[2] Recall K has class number 1.

commutative diagram

$$\begin{tikzcd}
V_{k,l} \arrow[r, "\Phi_d^{k,l}"] \arrow[d, "\widetilde{T}_\pi"] & (\mathcal{M}_2 \otimes V_{k,l})_{\Gamma_d} \arrow[d, "T_\pi"] \\
V_{k,l} \arrow[r, "\Phi_d^{k,l}"] & (\mathcal{M}_2 \otimes V_{k,l})_{\Gamma_d}
\end{tikzcd}$$

where the horizontal arrows are the Manin surjection $\Phi_d^{k,l}$, see (4.6). One can see this result as a transfer of the Hecke action from modular symbols to $V_{k,l}$. Alternatively we can say that when we equip $V_{k,l}$ with Heilbronn Hecke operators and modular symbols with the usual Hecke operators, the map $\Phi_d^{k,l}$ becomes *"Hecke equivariant"*.

The Hecke equivariance of $\Phi_d^{k,l}$ implies that $\ker(\Phi_d^{k,l})$ is stabilised by the Heilbronn Hecke operators (since the trivial subspace of the right hand side is Hecke stabilised). Thus the action descends to the quotient and we obtain an isomorphism

$$V_{k,l}/\ker(\Phi_d^{k,l}) \simeq (\mathcal{M}_2 \otimes_{\mathbb{C}} V_{k,l})_{\Gamma_d}$$

of Hecke modules.

4.4 Serre's conjecture

Serre's conjecture was originally formulated for Galois representations over $\mathbb{Q}$, in [Ser87]. However, since then it has been generalised to many more settings. Of interest to us is Serre's conjecture over imaginary quadratic fields, whose statement we take from [Tor12].

Let $\rho : G_K \to \mathrm{GL}(V)$ be a continuous 2-dimensional mod p representation, and write $\mathfrak{n}$ for its Serre conductor. Serre's conjecture predicts the existence of weights k, l and twists a, b in $\{0, \dots, p-1\}$, and a Dirichlet character χ of conductor $\mathfrak{n}$ (this is the character of Section 2.3.3) such that there is a Hecke eigenclass class $f \in H^i(\Gamma_0(\mathfrak{n}), V_{k,l}^{s,t,\chi}(\mathbb{F}_q))$, where $i = 1, 2$, corresponding to ρ in the following way:

for all primes $\mathfrak{p} \nmid p\mathfrak{n}$, we have

$$\mathrm{trace}(\rho(\mathrm{Frob}_{\mathfrak{p}})) = a_{\mathfrak{p}}(f),$$

where $a_{\mathfrak{p}}(f)$ is the eigenvalue of the $\mathfrak{p}^{th}$ Hecke operator $T_{\mathfrak{p}}$ when applied to f.

Much work has been done verifying Serre's conjecture for imaginary quadratic fields, see e.g. [Ş08], [Tor12]. We will assume the truth of the conjecture, often starting with a Galois representation ρ and finding the associated cohomology classes with the level, character and weight recipes in Section 2.3.

5. PERIOD POLYNOMIALS AND CONGRUENCES OF MODULAR FORMS

In this chapter, we define the space $W_{k,l}$ of *period polynomials of weight* (k, l) over an imaginary quadratic field K. This space is a subspace of $V_{k,l}$, the space of polynomials in homogeneous of degree k in X and Y and homogeneous of degree l in $\overline{X}$ and $\overline{Y}$. By relating period polynomials to modular symbols, we define an action of the Hecke algebra on $W_{k,l}$, allowing us to find explicit expressions for period polynomials attached to particular Bianchi Hecke eigenforms over K. We then perform certain computations on the polynomials in the space $W_{10,10}$ over $\mathbb{Q}(\sqrt{-11})$ to demonstrate methods for detecting and proving the existence of congruences of Bianchi modular forms.

5.1 Period polynomials

Let $\Gamma = \Gamma_d$ be one of the five Euclidean Bianchi groups. Consider the linear map

$$Z^1_{\text{par}}(\Gamma, V_{k,l}) \longrightarrow V_{k,l}, \qquad f \mapsto f(S) \tag{5.1}$$

where the left hand side is the space of *parabolic* 1-cocycles, defined in Section 4.2.2. The **space** $W_{k,l}$ **of period polynomials** of weight (k, l) for Γ is defined as the subspace of $V_{k,l}$ given by the image of this map.

In Section 5 of [Kar22], Karabulut gives explicit descriptions of the spaces of period polynomials for our Bianchi groups, in terms of kernels of elements of the group ring

$\mathbb{C}[\Gamma]$, which we now recall.

Remark 5.1.1. The paper of Karabulut [Kar22] actually defines the space of period polynomials in the parallel weight case, and only proves the defining equations for the space of period polynomials in that setting. However, the proofs themselves make no specific use of the parallel weight, and work exactly the same in the non-parallel weight case.

Recall in Section 4.1.5 we defined the matrices $S = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$, $T = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ and $T_\omega = \left(\begin{smallmatrix} 1 & \omega \\ 0 & 1 \end{smallmatrix}\right)$, where $\{1, \omega\}$ form a $\mathbb{Z}$-basis for $\mathbb{Z}_K$.

5.1.1 The case $d = 1$

In this case, we have

$$W_{k,l} = \ker(1+S) \cap \ker(1-L) \cap \ker(1+U+U^2) \cap \ker(1+E+E^2), \tag{5.2}$$

where $E = T_\omega S L = \left(\begin{smallmatrix} -1 & \omega \\ \omega & 0 \end{smallmatrix}\right)$.

5.1.2 The case $d = 2$

In this case, we have

$$W_{k,l} = \ker(1+S) \cap \ker(1+U+U^2) \cap \ker(1+ST_\omega + T_\omega S + T_\omega^{-1} S T_\omega S). \tag{5.3}$$

5.1.3 The case $d = 3$

In this case, we have

$$W_{k,l} = \ker(1+S) \cap \ker(1-L) \cap \ker(1+U+U^2) \cap \ker(1+E+E^2),$$

where $E = T^{-1} T_\omega S L = \left(\begin{smallmatrix} -1 & \omega-1 \\ \omega & 0 \end{smallmatrix}\right)$. As in Section 4.1.5.3, we note that this definition is slightly different to that found in [Kar22], again due to our choice of basis for $\mathcal{O}_3$.

5.1.4 The case $d = 7$

In this case, we have

$$W_{k,l} = \ker(1+S) \cap \ker(1+U+U^2) \cap \ker(T + T_\omega ST + ST_\omega^{-1}ST_\omega + ST_\omega).$$

5.1.5 The case $d = 11$

In this case we have

$$\begin{aligned} W_{k,l} = &\ker(1+S) \cap \ker(1+U+U^2) \cap \\ &\ker(T + T_\omega ST + ST_\omega^{-1}ST_\omega + ST_\omega + TT_\omega^{-1}ST_\omega ST + ST_\omega T^{-1}ST_\omega^{-1}ST_\omega). \end{aligned}$$

5.2 Period polynomials and modular symbols

We now discuss the relationship between modular symbols and period polynomials. More precisely, we will prove that for each of our Euclidean Bianchi groups,

$$W_{k,l} = \left(\ker(\Phi^{k,l})\right)^\perp$$

is the orthogonal complement of $\ker(\Phi^{k,l})$ under the pairing $\langle\cdot,\cdot\rangle$ (4.1). The proof in each case essentially boils down to checking certain relations (coming from the group algebra $\mathbb{C}[\Gamma_d]$) are satisfied. We will make copious use of the following basic fact from linear algebra:

Lemma 5.2.1. Let F be an algebraically closed field of characteristic p, V a finite-dimensional vector space over F, and g an F-endomorphism of V of finite order n, such that $\gcd(n, p) = 1$. Then

$$\operatorname{im}(1-g) = \ker(1 + g + \ldots + g^{n-1})$$

and

$$\ker(1-g) = \operatorname{im}(1+g+\ldots+g^{n-1}).$$

Remark 5.2.2. We use this result to relate spaces $W_{k,l}$ of period polynomials to cohomology groups. In the definitions of $W_{k,l}$ for each of the five fields we consider, there are matrices of orders 2 and 3, meaning this result cannot be used to guarantee an isomorphism in the cases of characteristic 2 and 3. In practice, we find that the dimensions and Hecke eigenvalue systems given by the spaces $W_{k,l}$ in these cases seem to still match those of H^2.

5.2.0.1 *The case $d = 1$*

Recall from (4.8) that the kernel of $\Phi_1^{k,l}$ is equal to $V_{k,l}^S + V_{k,l}^{SL} + V_{k,l}^U + V_{k,l}^E$ which we will denote $\mathbf{V}$ for compactness. We rewrite

$$\mathbf{V} = \ker(1-S) + \ker(1-SL) + \ker(1-U) + \ker(1-E),$$

and note that it follows from (4.2) that the kernel of T is orthogonal to the image of T^* for any element $T \in \mathbb{C}[\Gamma_1]$, and that $g^* = g^{-1}$ for $g \in \Gamma_d$. Therefore, we have

$$\mathbf{V}^\perp = \operatorname{im}(1-S^{-1}) \cap \operatorname{im}(1-(SL)^{-1}) \cap \operatorname{im}(1-U^{-1}) \cap \operatorname{im}(1-E^{-1}),$$

and using Lemma 5.2.1, we get

$$\mathbf{V}^\perp = \ker(1+S) \cap \ker(1+LS) \cap \ker(1+U+U^2) \cap \ker(1+E+E^2).$$

Recalling the explicit description (5.2) of $W_{k,l}$ as

$$W_{k,l} = \ker(1+S) \cap \ker(1-L) \cap \ker(1+U+U^2) \cap \ker(1+E+E^2),$$

we see that, to prove $W_{k,l} = \mathbf{V}^\perp$, we just need to prove that every $P \in W_{k,l}$ satisfies $P \cdot (1+LS) = 0$, and that every $Q \in \mathbf{V}^\perp$ satisfies $Q \cdot (1-L) = 0$.

Let $P \in W_{k,l}$, so $P = -P \cdot S$ and $P = P \cdot L$. Noting that $SLS = L$, we find

$$\begin{aligned} P \cdot (1 + LS) &= P + P \cdot LS \\ &= P - P \cdot SLS \\ &= P \cdot (1 - L) = 0. \end{aligned}$$

Hence, $W_{k,l} \subseteq \mathbf{V}^{\perp}$, and the same argument in reverse shows that $\mathbf{V}^{\perp} \subseteq W_{k,l}$.

5.2.0.2 The case $d = 2$

We proceed in the same way as the previous case. The kernel of Φ_2^k is $\mathbf{V} = V_{k,l}^S + V_{k,l}^U + V_{k,l}^A(1 - T_\omega^{-1})$, where $A = T_\omega^{-1} S T_\omega S$ has order 2. We want to use the same trick as before to compute the orthogonal complement, but in this case there is the extra factor of $1 - T_\omega^{-1}$ on the last term. So we rewrite $\ker(1 - A)$ as $\operatorname{im}(1 + A)$, to get

$$\mathbf{V} = \ker(1 - S) + \ker(1 - U) + \operatorname{im}(1 + A - T_\omega^{-1} - AT_\omega^{-1}). \tag{5.4}$$

Then, taking complements and turning images to kernels with Lemma 5.2.1, we get

$$\mathbf{V}^{\perp} = \ker(1 + S) \cap \ker(1 + U + U^2) \cap \ker(1 + A - T_\omega - T_\omega A).$$

Again recalling the explicit description of $W_{k,l}$ in (5.3), we see that proving $W_{k,l} = \mathbf{V}^{\perp}$ amounts to showing that $P \in W_{k,l}$ satisfies $P \cdot (1 - T_\omega + T_\omega^{-1} S T_\omega S - S T_\omega S) = 0$, and that $Q \in \mathbf{V}^{\perp}$ satisfies $Q \cdot (1 + S T_\omega + T_\omega^{-1} S T_\omega S + T_\omega S) = 0$.

Since $P \cdot (1 + S) = 0$, we can expand to get

$$\begin{aligned} P \cdot (1 - T_\omega + S T_\omega^{-1} S T_\omega - S T_\omega S) &= P - P \cdot T_\omega + P \cdot S T_\omega^{-1} S T_\omega - P \cdot S T_\omega S \\ &= P + P \cdot S T_\omega + P \cdot S T_\omega^{-1} S T_\omega + P \cdot T_\omega S \\ &= P \cdot (1 + S T_\omega + S T_\omega^{-1} S T_\omega + T_\omega S) = 0, \end{aligned}$$

and, since $Q \cdot (1 + S) = 0$ also, the same logic works in reverse to give $W_{k,l} = \mathbf{V}^{\perp}$.

5.2.0.3 The case $d = 3$

Proceeding as before, writing $\mathbf{V} = \ker(\Phi_3^{k,l})$, we get

$$\mathbf{V}^\perp = \ker(1 + LS) \cap \ker(1 + SL) \cap \ker(1 + U + U^2).$$

First, we show $W_{k,l} \subseteq \mathbf{V}^\perp$. This amounts to showing $P\cdot(1+SL) = P\cdot(1+LS) = 0$ for all $P \in W_{k,l}$. To prove the first identity, we note that $P\cdot(1+S) = 0$, so $P\cdot(1+S)L = 0$. Expanding and using that $P = P \cdot L$, we get $0 = P + P \cdot SL = P \cdot (1 + SL)$. The second is proved in the same way, swapping the roles of S and L.

To show $\mathbf{V}^\perp \subseteq W_{k,l}$, we must show

$$Q \cdot (1 + S) = Q \cdot (1 - L) = Q \cdot (1 + E + E^2) = 0,$$

for $Q \in \mathbf{V}^\perp$. If Q *is* in $\mathbf{V}^\perp$, it satisfies $Q \cdot SL = Q \cdot LS$, and since $LSL = S$ and $SL^2 = LS$, we have

$$\begin{aligned} Q \cdot (1 + S) &= Q \cdot (1 + LSL) \\ &= Q + (Q \cdot LS) \cdot L \\ &= Q + Q \cdot SL^2 = Q \cdot (1 + LS) = 0. \end{aligned}$$

Using this, we also obtain

$$\begin{aligned} Q \cdot (1 - L) &= Q - Q \cdot L \\ &= Q + Q \cdot SL = Q \cdot (1 + SL) = 0. \end{aligned}$$

Finally, we show $Q \in \ker(1 + E + E^2)$, where $E = T^{-1}T_\omega SL$. A quick computation shows E can also be written $LTLS$, and $E^2 = LTSTLS$. Using $Q = Q \cdot L$, we get

$$\begin{aligned} Q \cdot (1 + E + E^2) &= Q + Q \cdot LTLS + Q \cdot LTSTLS \\ &= Q + Q \cdot TLS + Q \cdot (TS)TLS. \end{aligned}$$

Now we use that $Q \cdot TS = -Q - Q \cdot TSTS$ to get

$$Q \cdot (1 + E + E^2) = Q - Q \cdot TSTSTLS.$$

Now, using that $(ST)^3 = 1$, as well as a final use of $Q = -Q \cdot S$, we get

$$\begin{aligned} Q \cdot (1 + E + E^2) &= Q + Q \cdot STSTSTLS \\ &= Q + Q \cdot LS = Q \cdot (1 + LS) = 0. \end{aligned}$$

5.2.0.4 The case $d = 7$

As before, write $\mathbf{V} = \ker(\Phi_7^{k,l})$. We have

$$\mathbf{V}^\perp = \ker(1 + S) \cap \ker(1 + U + U^2) \cap \ker(1 + T_\omega S + ST_\omega^{-1} ST_\omega T^{-1} + ST_\omega T^{-1}),$$

with the final kernel coming from $(V_{k,l}^A(1 + g))^\perp = \ker((1 + g^{-1})(1 + A))$, in the same manner as (5.4). Here $A = ST_\omega^{-1} ST_\omega T^{-1}$ and $g = ST_\omega^{-1}$.

The only difference between $\mathbf{V}^\perp$ and $W_{k,l}$ is the third kernel of each space. For $W_{k,l}$, this is $\ker(T + T_\omega ST + ST_\omega^{-1} ST_\omega + ST_\omega)$. But T acts on $V_{k,l}$ injectively, so the two kernels are equal.

5.2.0.5 The case $d = 11$

We have

$$\begin{aligned} \mathbf{V}^\perp = &\ker(1 + S) \cap \ker(1 + U + U^2) \cap \\ &\ker(1 + T_\omega S + ST_\omega^{-1} ST_\omega T^{-1} + ST_\omega T^{-1} + TT_\omega^{-1} ST_\omega S + ST_\omega T^{-1} ST_\omega^{-1} ST_\omega T^{-1}). \end{aligned}$$

Again, the final kernel comes from $(V_{k,l}^A(1 + g))^\perp$. In this case it is equal to $\ker((1 + g^{-1})(1 + A + A^2))$, where $A = ST_\omega^{-1} ST_\omega T^{-1}$ and $g = ST_\omega^{-1}$. Comparing to the corresponding kernel for $W_{k,l}$, which is

$$\ker(T + T_\omega ST + ST_\omega^{-1}ST_\omega + ST_\omega + TT_\omega^{-1}ST_\omega ST + ST_\omega T^{-1}ST_\omega^{-1}ST_\omega),$$

we again see the equality $\mathbf{V}^\perp = W_{k,l}$ from the injectivity of T.

5.2.1 Hecke operators on period polynomials

Recall that in Section 5.2, we showed that the space of period polynomials $W_{k,l}$ is the orthogonal complement of the kernel of the Manin surjection:

$$W_{k,l} = \ker(\Phi^{k,l})^\perp.$$

with respect to the symmetric bilinear form (4.1). Note that as (4.1) is non-degenerate, we have

$$V_{k,l} = W_{k,l} \oplus \ker(\Phi^{k,l}). \tag{5.5}$$

Recall that for $g \in \mathrm{GL}_2(\mathbb{C})$ and $v, w \in V_{k,l}$, we have

$$\langle v \cdot g, w\rangle = \langle v, w \cdot g^\iota\rangle$$

with $g^\iota = \det(g)g^{-1}$. Therefore, for a Heilbronn Hecke operator $\widetilde{T}_\pi$, we have

$$\langle v \cdot \widetilde{T}_\pi, w\rangle = \langle v, w \cdot \widetilde{T}_\pi^*\rangle \tag{5.6}$$

where the operator $\widetilde{T}_\pi^*$ is given as

$$w \cdot \widetilde{T}_\pi^* := \sum_{g \in H_\mathfrak{p}} w \cdot g^\iota. \tag{5.7}$$

Since the Heilbronn Hecke operators $\widetilde{T}_\pi$ stabilise the kernel of $\Phi^{k,l}$ and the space $W_{k,l}$ is orthogonal to this kernel, it follows from (5.6) that the *adjoint operators* $\widetilde{T}_\pi^*$ stabilise $W_{k,l}$. We record this discussion within the following corollary.

Corollary 5.2.3. For the Euclidean Bianchi groups, the space $W_{k,l}$ is equipped with an action of the adjoint Heilbronn Hecke operators as in (5.7). The canonical isomorphism

$$\phi : W_{k,l} \simeq V_{k,l}/\ker \Phi_d^{k,l}$$

arising from the orthogonal decomposition (5.5) is Hecke equivariant, that is,

$$\phi(w \cdot \widetilde{T}_\pi^*) = \phi(w) \cdot \widetilde{T}_\pi$$

for any prime element $\pi \in \mathcal{O}_d$.

5.3 Choosing a period of significance

Let $f \in H^1(\Gamma_0(\mathfrak{n}), V)$ for some weight module V. In this section, we speculate on how to choose and compute a period of f that "ought" to capture information about the ranks of Selmer groups of representations ρ_f.

5.3.1 The classical case (trivial weight)

In the classical case, the Birch & Swinnerton-Dyer conjecture suggests that the rank of an elliptic curve $E/\mathbb{Q}$ (and so the rank of its p-Selmer group) should be captured by the vanishing of the special L-value $L(E,1)$ (see Section C.16 of [Sil09]). In cohomology, there is a Hecke eigenvector $[f_E] \in H^1(\Gamma_0(N), \mathbb{C})$, where N is the conductor of E, such that $\langle f, \{0, \infty\}\rangle$ is this L-value (up to some scalar factor depending on the choice of eigenvector f_E).

This pairing can be computed explicitly using a "closing the path trick" (see [Cre97], Section 2.8), which amounts to averaging the cocycle f_E (really just a homomorphism $\Gamma_0(N) \to \mathbb{C}$) over certain matrices. Let $q \nmid N$ be a prime. Then

$$\langle f, \{0, \infty\}\rangle = \frac{1}{1 + q - a_q} \sum_{k=0}^{q-1} f(M_k),$$

where $M_k \in \Gamma_0(N)$ is a matrix taking 0 to k/q under fractional linear transformation. This value is independent of the choice of cocycle representing the class f, as the sum vanishes on coboundaries. We call this value the **period average of** f.

When $\mathbb{C}$ is replaced with $\mathbb{F}_p$, the same computation works and again gives a well-defined output. The only extra consideration to make is that $1 + q \not\equiv a_q \pmod{p}$, which ensures we can invert $1 + q - a_q$.

We denote the value of this average for $f \in H^1(\Gamma_0(N), \mathbb{C})$ by $\mathcal{P}(f)$, as the value is independent of the prime q chosen to compute it.

To compute the reduction of f_E modulo p, it suffices to find an eigenvector $\tilde{f}_E$ in $H^1(\Gamma_0(N), \mathbb{Z})$ whose Hecke eigenvalues match those of f_E. This integral cocycle class is simply f_E, scaled by some period of E. Then this integral class can be reduced modulo a prime p to get $\overline{f}_E \in H^1(\Gamma_0(N), \mathbb{F}_p)$.

Not all classes can be reduced mod p without affecting the period information. For example, let E be the elliptic curve

$$E \colon y^2 + y = x^3 - x^2.$$

This is the curve 11.a3. The associated Hecke eigenform has eigenvalues

q	2	3	5	7	11	13	17
a_q	-2	-1	1	-2	1	4	-2

Tab. 5.1: Eigenvalues of the unique cuspidal modular form of weight 2, level $\Gamma_0(11)$.

and one can prove that $a_q \equiv 1 + q \pmod{5}$ for all primes $q \nmid 11$. Thus the period average of $\overline{f}_E \in H^1(\Gamma_0(11), \mathbb{F}_5)$ cannot be defined for any choice of the prime q.

It should be clear that, if $\mathcal{P}(f_E) = 0$, then $\mathcal{P}(\overline{f}_E) = 0$ as well (when it is defined). However, it is not always the case that a non-zero period stays non-zero on reduction modulo p, see Example 5.3.1 below.

It is also possible to compute this period without averaging. The space of period polynomials for the trivial weight module V_0 and level N is just a subspace of $\mathbb{C}[\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})]$; for non-trivial weight is is a subspace of $\mathbb{C}[\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})] \otimes V_k$. The period polynomial of a classical weight k cusp form f is a polynomial with coefficients expressed in terms of the critical L-values of f. These are obtained by integrals of $f(z)z^i$ for $i \in \{0, \dots, k-2\}$. In [PP13], the vector representing f in the space of period polynomials of level N and weight k is called the *extended period polynomial*; we use the same terminology here in the Bianchi case. The coefficients of the extended period polynomial are again obtained by integrals of $f(z)z^i$, but for each element $\gamma = (c : d) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$, f is replaced with $f(\gamma \cdot z)(cz+d)^-k$. Here we are implicitly treating γ as a coset representative in $\Gamma_0(N)\backslash\Gamma$.

In Section 5.5 of that paper, a numerical example is given, showing how the classical period polynomial of a weight k modular form can be extracted from its extended period polynomial in cohomology. This amounts to restricting to $(0 : 1) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$, which corresponds to the identity coset in $\Gamma_0(N)\backslash\Gamma$. The passage from the extended period polynomial to the classical one can then be viewed as the evaluation map of the Shapiro lemma.

5.3.2 The Bianchi case (trivial weight)

The analogous theory still holds in the case of our Euclidean imaginary quadratic fields, see [CW94], Section 2.8. In particular, we replace the level N with an ideal $\mathfrak{n} \lhd \mathcal{O}_K$ and the prime q with a prime ideal $\mathfrak{q} \nmid \mathfrak{n}$ to obtain

$$\mathcal{P}(F) = \langle F, \{0, \infty\} \rangle = \frac{1}{1 + N(\mathfrak{q}) - a_{\mathfrak{q}}} \sum_{\alpha \pmod{\mathfrak{q}}} F(M_\alpha),$$

where F is now a Bianchi form and M_α takes 0 to α/π (where π generates $\mathfrak{q}$) under the action of Equation (4.1.3). As in the classical case, it is also possible to compute $\mathcal{P}(F)$ without averaging, by instead computing the space of period polynomials, and identifying the period polynomial r_F (defined in Equation (5.8)) by its eigenvalues. Then $\mathcal{P}(F)$ is again the coefficient of r_F coming from the identity component $(0:1)$ of the projective line $\mathbb{P}^1(\mathcal{O}_K/\mathfrak{n})$.

Since either method involves a choice of generator for the 1-dimensional space of either F or r_F, we cannot pin down an exact canonical value for $\mathcal{P}(F)$ without more information. However, we are only really interested in whether the period vanishes or not, and scaling has no effect on this. Even for mod p classes, the only way to scale F or r_F to be 0 is by multiplying by p, which kills the whole class anyway.

Example 5.3.1. Let $K = K_3 = \mathbb{Q}(\omega_3)$, where recall $\omega_3 = \omega = \frac{1+\sqrt{-3}}{2}$, and let $\mathfrak{n} = (8\omega - 16)$ (norm 192). There is one form in $F \in H^1(\Gamma_0(\mathfrak{n}), \mathbb{C})$, corresponding to the Bianchi modular form 192.1-a. Its Hecke eigenvalues begin

$\mathfrak{p}$	$(\omega-2)$	$(\omega-3)$	$(\omega+3)$	$(\omega-4)$	$(-2\omega+5)$	$(2\omega+3)$	(5)
$N(\mathfrak{p})$	7	7	13	13	19	19	25
$a_{\mathfrak{p}}$	0	0	-2	-2	-4	-4	-6

Tab. 5.2: Hecke eigenvalues of the unique Bianchi modular form over $\mathbb{Q}(\sqrt{-3})$ of weight 2 and level $(8\omega - 16)$.

The exact value of its period average depends on the choice of scaling, but in particular, it does not vanish. The form F generates a 1-dimensional subspace of $H^1(\Gamma_0(\mathfrak{n}), \mathbb{C})$; if we choose the integral scaling (meaning we take a generator $\tilde{F}$ of $\langle F\rangle \cap H^1(\Gamma_0(\mathfrak{n}), \mathbb{Z}_K)$), we find $\mathcal{P}(\tilde{F}) = \frac{3}{4}$.

Meanwhile, there are three linearly independent classes in $H^1(\Gamma_0(\mathfrak{n}), \mathbb{F}_3)$. By computing eigenvalue systems, we find a multiplicity 1 class whose eigenvalues are the reduction modulo 3 of F. We compute its period average to be 0. In some sense we expect this result, as the reduction mod 3 map is defined

$$H^1(\Gamma_0(\mathfrak{n}), \mathbb{Z}_K) \to H^1(\Gamma_0(\mathfrak{n}), \mathbb{F}_3)$$

and $\mathcal{P}(\tilde{F})$ has a 3 in its numerator.

5.3.3 Non-trivial weight

The choice of period is trickier in the higher weight case; for f a modular form over $\mathbb{Q}$, the pairing $\langle f, \{0, \infty\}\rangle$ gives an algebraic period polynomial, rather than a single value. When the weight of f is even, the usual choice, as in for example [DSW04], is to take the middle coefficient of the period polynomial.

This becomes more complicated for Bianchi modular forms, as the period polynomial now has two degrees of freedom, coming from the pairs of variables X^iY^{k-i} and $\overline{X}^j\overline{Y}^{l-j}$. In characteristic 0, the only "interesting" examples come from parallel weight, i.e. $k = l$, and when k is even there is again a middle coefficient, coming from the $X^{k/2}Y^{k/2}\overline{X}^{k/2}\overline{Y}^{k/2}$ term. See, for example, Figure 5.2, an example of a weight $(10, 10)$ period polynomial, whose central coefficient vanishes. In fact, all the coefficients along the main anti-diagonal vanish.

In characteristics bigger than 0, interesting period polynomials can come from non-parallel weights, making the question of exactly which coefficient, if any, should carry the interesting information unclear. In examples, we simply note the whole period polynomial, and leave the question of which coefficients are significant alone.

5.4 Congruences of period polynomials

In this section, we report on our numerical investigations into the Hecke module structure of some spaces of Bianchi period polynomials. From the point of view of number theory, it is better to work with the group $\mathrm{PGL}_2(\mathcal{O})$, as only then the Hecke operators associated to the prime elements π and $u\pi$ are the same for any unit $u \in \mathcal{O}^\times$, allowing one to associate Hecke operators to prime ideals. From the perspective of period polynomials, this amounts to computing the plus-subspaces that we now define.

5.4.1 *The plus-space*

Let ε be a generator of the group $\mathcal{O}_d^\times$. The element $J = \left(\begin{smallmatrix} \varepsilon & 0 \\ 0 & 1 \end{smallmatrix}\right)$ normalises Γ and thus gives rise to an involution[1] δ on cohomology and homology groups of Γ which is well-known to commute with the Hecke operators. The involution δ on a 1-cocycle $f \in H^1(\Gamma, V_{k,l}(\mathbb{C}))$ is given by

$$\delta(f)(g) = f(JgJ^{-1}) \cdot J.$$

Thus f takes values in $V_{k,l}(\mathbb{C})$, and the action of J is the usual action defined previously in 4.1.2. When $\varepsilon = -1$, we have that $JSJ^{-1} = S$, and so the map (5.1) tells us that the corresponding involution on the space of Bianchi period polynomials is given by

$$\delta(P)(X, Y, \overline{X}, \overline{Y}) = P(X, Y, \overline{X}, \overline{Y}) \cdot J = P(-X, Y, -\overline{X}, \overline{Y}).$$

When $\varepsilon \neq -1$ it is less obvious that δ is an involution, although it is—see the comment in Section 2.4 of [CW94] for details. We define the **plus-space** $W_{k,l}^+$ of Bianchi period polynomials to be the fixed subspace of the involution δ on $W_{k,l}$:

$$W_{k,l}^+ \coloneqq \{P \in W_{k,l} \mid \delta(P) = P\}.$$

Notice that, as the involution δ on the space of Bianchi period polynomials commutes with the action of Heilbronn Hecke operators, the plus-space $W_{k,l}^+$ is stabilised under the Hecke action. For the rest of the chapter, we will assume $k = l$.

In [FGT10], the authors verified that, for the five Euclidean Bianchi groups, among all the spaces $S_{k+2}(\mathrm{PGL}_2(\mathcal{O}_d))$ of cuspidal Bianchi modular forms of full level $\mathrm{PGL}_2(\mathcal{O}_d)$ and weight k within the scope given in the below table

[1] This involution is actually nothing but the Hecke operator associated to the double coset of J.

$-d$	1	2	3	7	11
$k \leq$	104	141	116	132	153

Tab. 5.3: Ranges of weights computed in [FGT10].

there is a unique space which contains non-base-change ("genuine") Bianchi modular forms; $S_{12}(\mathrm{PGL}_2(\mathcal{O}_{11}))$ is a 3-dimensional space, containing a pair F_1, F_2 of Galois conjugate genuine forms, alongside the base-change lift of the classical discriminant modular form Δ.

In this section, using computer programs we developed to compute the spaces of Euclidean Bianchi period polynomials and the action of the associated Heilbronn Hecke operators, we exhibit two different congruences concerning the genuine Bianchi cuspforms living in $S_{12}(\mathrm{PGL}_2(\mathcal{O}_{11}))$; in the first, F_1, F_2 are congruent to (the base-change lift of) an Eisenstein series E_{12} mod 173, and in the second, they are congruent to Δ mod 43. We summarise these congruences, along with a congruence between the base-change lifts of Δ and E_{12}, in the below congruence graph, with vertices representing forms and edges representing moduli:

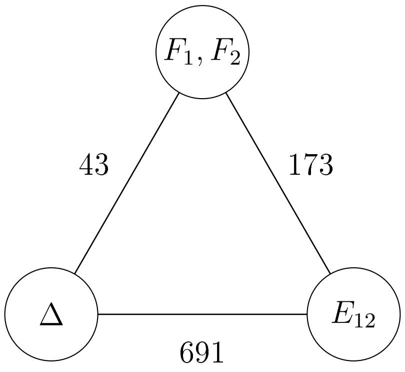

Fig. 5.1: Congruence graph of Hecke eigenvalue systems in $S_{12}(\mathrm{PGL}_2(\mathcal{O}_{11}))$.

The congruences involving F_1 and F_2 are, as far as we know, the first congruences in the literature between higher weight genuine cusp forms and base-change cusp forms, Eisenstein series. A novelty concerning these congruences is that we "detect" them using Bianchi period polynomials, as we describe below.

We will often refer to Hecke eigenvectors in the space of Bianchi period polynomials as Bianchi modular forms. This is justified under the embeddings discussed in the introduction and our results concerning the Hecke action.

5.4.2 *Cusp forms and algebraicity*

Let us start the discussion by summarising some algebraicity results for our Bianchi period polynomials which follow from well-known results of Hida, see Sections 3, 6 and 8 of [Hid94].

Let $f \in S_{k+2}(\mathrm{PGL}_2(\mathcal{O}))$ be a level 1 Bianchi cusp form of weight $k+2$. As proven by Harder, and explicated by Hida, there is a $V_{k,k}(\mathbb{C})$-valued harmonic differential 1-form ω_f on the arithmetic 3-fold X_Γ (given by the orbit space of Γ on the hyperbolic 3-space $\mathcal{H}_3 = \mathbb{C} \times \mathbb{R}^+$). The Eichler-Shimura map Θ mentioned in Section 4.1.4 takes f to the class of the 1-cocycle

$$\Theta(f) : \Gamma \to V_{k,k}(\mathbb{C}), \qquad \gamma \mapsto \int_z^{\gamma \cdot z} \omega_f$$

by integrating the form ω_f over the geodesic from z to $\gamma \cdot z$. Here z is a fixed point in $\mathcal{H}$ or in the rational boundary $\mathbb{P}^1(K) = K \cup \{\infty\}$ of $\mathcal{H}_3$. Changing the point z changes the cocycle by a coboundary. Note that, if we choose $z = \infty$, the cocycle we obtain is parabolic.

Using the "evaluation at S" map (5.1), we can obtain a period polynomial from a parabolic cocycle. The **canonical**[2] **period polynomial** associated to $f \in S_{k+2}(\mathrm{PGL}_2(\mathcal{O}))$ is defined as

$$r_f := \int_0^\infty \omega_f \in W_{k,k}^+(\mathbb{C}). \tag{5.8}$$

[2] This polynomial is canonical because ω_f is canonical once f is appropriately scaled.

Remark 5.4.1. As in the case of classical cusp forms, the coefficients of the canonical period polynomial of f are related to the special values of the L-function of f: the critical values $L(f, s+1)$ with $0 \leq s \leq k$ appear in the coefficients of the diagonal terms $X^{k-s}Y^s\overline{X}^{k-s}\overline{Y}^s$, whereas off-diagonal terms are related to values of the twisted L-functions of f, see Theorem 2.11 of [Wil17] for details.

Now assume that f is a simultaneous eigenvector for all the Hecke operators. It is a classical result that the field extension obtained by adjoining all the Hecke eigenvalues of f to $\mathbb{Q}$ is a number field which we will denote by $\mathbb{Q}(f)$. We will also use the field $K(f) := K\mathbb{Q}(f)$, which is the field over which period polynomials over K are defined. It follows from results of Hida that there is a finite extension $F/K(f)$ and a complex period $\Omega_f \in \mathbb{C}^\times$ such that

$$\tfrac{1}{\Omega_f} r_f \in W_{k,k}(F)^+. \tag{5.9}$$

Our computer programs compute the space $W_{k,k}(K)^+$, which is stabilised by Hecke operators. Since we have

$$W^+_{k,k}(K) \otimes_K \mathbb{C} \simeq W^+_{k,k}(\mathbb{C}),$$

our Hecke operators capture all of the Hecke module information of $W^+_{k,k}(\mathbb{C})$, and hence of $S_{k+2}(\mathrm{PGL}_2(\mathcal{O}))$. By suitably replacing the coefficient field K with a finite extension (coming from the characteristic polynomials of the Hecke operators) we obtain **algebraic** period polynomials (spanning a K-line) which realise the Hecke eigenvalue systems that we observe.

5.4.3 Eisenstein series

It is important to note that the space $S_{k+2}(\mathrm{PGL}_2(\mathcal{O}))$ does not account for all of $W^+_{k,k}(\mathbb{C})$; there is a one-dimensional complement which "sees" an Eisenstein series. Here is one way to explain this.

In Section 5.2 we established that there is a Hecke equivariant isomorphism

$$W_{k,k}^{+}(\mathbb{C}) \simeq H^2(\Gamma, V_{k,k}(\mathbb{C}));$$

in turn, it is well-known that there is also a Hecke equivariant isomorphism

$$H^2(\Gamma, V_{k,k}(\mathbb{C})) \simeq S_{k+2}(\Gamma) \oplus \mathrm{Eis}_{k+2}(\Gamma),$$

where, for $k > 0$, the space Eis_{k+2} is the one-dimensional[3] space generated by the Eisenstein series E_{k+2} of weight $k+2$ associated to the single cusp (since K has class number 1) of the arithmetic hyperbolic 3-fold X_Γ. This is a consequence of the relation

$$H^2_{\mathrm{par}}(\Gamma, V_{k,k}(\mathbb{C})) \simeq S_{k+2}(\Gamma),$$

appearing in Section 5.1 of [Gha99] (albeit in the form of cuspidal cohomology of $\Gamma\backslash\mathcal{H}_3$), combined with the above observation that the single Eisenstein series for weight $k+2 > 2$ comes from the single cusp of $\Gamma\backslash\mathcal{H}_3$.

We therefore see that there is a one-dimensional complement to the image of $S_{k+2}(\Gamma)$ in $W_{k,k}^{+}(\mathbb{C})$, coming from E_{k+2} and realising the eigenvalue system $\{N(\mathfrak{p})^{k+1}+1\}$. One can show that this one-dimensional complement is spanned by the period polynomial $X^k\overline{X}^k - Y^k\overline{Y}^k$, corresponding to the image of the one-dimensional space of parabolic coboundaries $B^1_{\mathrm{par}}(\Gamma, V_{k,k}(\mathbb{C})) \subset Z^1_{\mathrm{par}}(\Gamma, V_{k,k}(\mathbb{C})) \to W_{k,k}(\mathbb{C})$.

This is the Bianchi counterpart of the classical period polynomial $X^k - Y^k$, which is well-known (going back to [Man73], see also Theorem 1 of [KZ84]) to realise the Eisenstein eigenvalue system $\{p^{k+1}+1\}$. We will not prove this as we do not need it; however we will see that it holds in the specific Bianchi period polynomial space that we will compute with in the rest of the chapter.

[3] For $k = 0$, the space is trivial, see Proposition 1 of [Rc13].

5.4.4 Congruences of period polynomials

Two algebraic period polynomials r_f and r_g each have their own field of definition, $K(f)$ and $K(g)$ respectively. Both polynomials can be defined over the compositum of these fields, $K(f,g) := K(f)K(g)$, and can therefore be scaled to have coprime[4] coefficients in the ring of integers $\mathcal{O}_{f,g}$ of $K(f,g)$. We write R_f and R_g for the resulting integral polynomials. For any prime ideal $\mathfrak{q}$ of $\mathcal{O}_{f,g}$, there is a mod $\mathfrak{q}$ reduction map

$$W_{k,k}(\mathcal{O}_{f,g}) \to W_{k,k}(\mathbb{F}_{\mathfrak{q}})$$

induced by the map reduction mod $\mathfrak{q}$ map $\mathcal{O}_{f,g} \to \mathbb{F}_{\mathfrak{q}} \simeq \mathcal{O}_{f,g}/\mathfrak{q}$. The polynomials R_f and R_g are **congruent mod $\mathfrak{q}$** if their images in $W_{k,k}(\mathbb{F}_{\mathfrak{q}})$ are equal. When two polynomials are congruent and neither of their images is the zero polynomial in $W_{k,k}(\mathbb{F}_{\mathfrak{q}})$, their Hecke eigenvalues are automatically congruent modulo $\mathfrak{q}$, as the reduction map is Hecke equivariant. We will use this fact to prove the existence of congruences between Bianchi modular forms of level 1 in the rest of this section.

Remark 5.4.2. In Sections 5.4.7, 5.4.8, we show that two integral polynomials are congruent modulo a prime $\mathfrak{q}$ in a number field L', and use this to conclude that Hecke eigenvalues are congruent modulo a prime $\mathfrak{p}$ in a subfield L. In this instance, it happens that the primes $\mathfrak{p}$ in L are inert in L', and since the Hecke eigenvalues live in L, we can conclude that the eigenvalues are congruent modulo a prime in L, rather than L' as the above would suggest.

5.4.5 The space

For the remainder of this section, we set $K = \mathbb{Q}(\sqrt{-11})$ and $\Gamma = \mathrm{PGL}_2(\mathcal{O}_{11})$. We compute that the space

$$W := W^+_{10,10}(K) \otimes_K \mathbb{C} \simeq W^+_{10,10}(\mathbb{C})$$

[4] Meaning the ideals generated by the coefficients are coprime.

is 4-dimensional using the code in B, agreeing with the computations of [FGT10]. Using the Hecke action, we find W consists of: the base-change of the classical cusp form Δ, which we still denote by Δ, the Eisenstein series E_{12}, which is the base-change of the classical level 1 weight 12 Eisenstein series, and two genuine Bianchi cusp forms F_1, F_2. The genuine forms have Hecke eigenvalues in the field $L = \mathbb{Q}(\beta)$, where $\beta = \sqrt{81829}$, and are conjugate by the non-trivial automorphism of L. We note that the primes that ramify in L are exactly $\{11, 43, 173\}$. We record the Hecke eigenvalues of the forms for the first few primes of K in Figure 5.4. Note that the Hecke eigenvalues for the genuine forms agree with those computed in Section 6.2.2 of [FGT10].

$\mathfrak{p}$	ω	$1-\omega$	2	$1+\omega$	$2-\omega$
$N(\mathfrak{p})$	3	3	4	5	5
$a_{\mathfrak{p}}(\Delta)$	252	252	-3250	4830	4830
$a_{\mathfrak{p}}(E_{12})$	177148	177148	4194305	48828126	48828126
$a_{\mathfrak{p}}(F_1)$	$\beta - 350$	$-\beta - 350$	-80	$26\beta - 5103$	$-26\beta - 5103$
$a_{\mathfrak{p}}(F_2)$	$-\beta - 350$	$\beta - 350$	-80	$-26\beta - 5103$	$26\beta - 5103$

Tab. 5.4: Hecke eigenvalue systems captured in $S_{12}(\mathrm{PGL}_2(\mathcal{O}_{11}))$.

5.4.6 Congruence between Δ and E_{12}

It immediately follows from the recipe[5] describing the behaviour of Hecke eigenvalues under base-change lifting that the famous congruence mod 691 (see e.g. [Man73]) between the classical modular forms Δ and E_{12} continues to hold between their base-change lifts.

The eigenvalue system associated to E_{12} cuts out a one-dimensional subspace of $W^+_{10,10}(K)$ which we see to be spanned by $X^{10}\overline{X}^{10} - Y^{10}\overline{Y}^{10}$.

Now consider the one-dimensional subspace of $W^+_{10,10}(K)$ cut out by (the eigenvalue

[5] Which can be found, e.g. here
https://www.lmfdb.org/knowledge/show/mf.bianchi.base_change

system associated to) Δ. We pick any non-zero Bianchi period polynomial in this one-dimensional subspace, and scale it[6] so that the first coefficient (i.e. the coefficient of the monomial $X^{10}\overline{X}^{10}$) is 1. Let us call this polynomial P and write

$$P = \sum_{0 \le i,j \le k} c_{i,j} X^{10-i} Y^i \overline{X}^{10-j} \overline{Y}^j.$$

We represent P as an 11×11 coefficient matrix $(c_{i,j})_{0 \le i,j \le 10}$:

$$\begin{pmatrix}
1 & 0 & \frac{55280}{31203} & 0 & \frac{5528}{10401} & 0 & \frac{3455}{55472} & 0 & \frac{691}{218421} & 0 & 0 \\
0 & -\frac{7082750}{1965789} & 0 & -\frac{193480}{93609} & 0 & -\frac{34550}{93609} & 0 & -\frac{65645}{2246616} & 0 & 0 & 0 \\
\frac{55280}{31203} & 0 & \frac{4892971}{1497744} & 0 & \frac{79465}{93609} & 0 & \frac{8983}{93609} & 0 & 0 & 0 & -\frac{691}{218421} \\
0 & -\frac{193480}{93609} & 0 & -\frac{21044405}{15726312} & 0 & -\frac{13820}{93609} & 0 & 0 & 0 & \frac{65645}{2246616} & 0 \\
\frac{5528}{10401} & 0 & \frac{79465}{93609} & 0 & \frac{156857}{499248} & 0 & 0 & 0 & -\frac{8983}{93609} & 0 & -\frac{3455}{55472} \\
0 & -\frac{34550}{93609} & 0 & -\frac{13820}{93609} & 0 & 0 & 0 & \frac{13820}{93609} & 0 & \frac{34550}{93609} & 0 \\
\frac{3455}{55472} & 0 & \frac{8983}{93609} & 0 & 0 & 0 & -\frac{156857}{499248} & 0 & -\frac{79465}{93609} & 0 & -\frac{5528}{10401} \\
0 & -\frac{65645}{2246616} & 0 & 0 & 0 & \frac{13820}{93609} & 0 & \frac{21044405}{15726312} & 0 & \frac{193480}{93609} & 0 \\
\frac{691}{218421} & 0 & 0 & 0 & -\frac{8983}{93609} & 0 & -\frac{79465}{93609} & 0 & -\frac{4892971}{1497744} & 0 & -\frac{55280}{31203} \\
0 & 0 & 0 & \frac{65645}{2246616} & 0 & \frac{34550}{93609} & 0 & \frac{193480}{93609} & 0 & \frac{7082750}{1965789} & 0 \\
0 & 0 & -\frac{691}{218421} & 0 & -\frac{3455}{55472} & 0 & -\frac{5528}{10401} & 0 & -\frac{55280}{31203} & 0 & -1
\end{pmatrix}$$

Guided by our desire to find a congruence with E_{12}, we let D denote the gcd of all the rational numbers that appear as "middle coefficients" (i.e. all the coefficients except for the first $(i,j) = (0,0)$ and the last $(i,j) = (10,10)$). That is,

$$D = \frac{\gcd\big(\{\text{numerator}(c_{i,j}) \mid (i,j) \neq (0,0), (10,10)\}\big)}{\operatorname{lcm}\big(\{\text{denominator}(c_{i,j}) \mid (i,j) \neq (0,0), (10,10)\}\big)} = \frac{691}{31452624},$$

so that when we scale P by $1/D$, the middle coefficients all become integers with overall gcd 1, giving the scaled period polynomial

[6] In fact, Magma by default gives us an eigenvector whose leading coefficient is 1.

$$\begin{pmatrix}
\frac{31452624}{691} & 0 & 80640 & 0 & 24192 & 0 & 2835 & 0 & 144 & 0 & 0 \\
0 & -164000 & 0 & -94080 & 0 & -16800 & 0 & -1330 & 0 & 0 & 0 \\
80640 & 0 & 148701 & 0 & 38640 & 0 & 4368 & 0 & 0 & 0 & -144 \\
0 & -94080 & 0 & -60910 & 0 & -6720 & 0 & 0 & 0 & 1330 & 0 \\
24192 & 0 & 38640 & 0 & 14301 & 0 & 0 & 0 & -4368 & 0 & -2835 \\
0 & -16800 & 0 & -6720 & 0 & 0 & 0 & 6720 & 0 & 16800 & 0 \\
2835 & 0 & 4368 & 0 & 0 & 0 & -14301 & 0 & -38640 & 0 & -24192 \\
0 & -1330 & 0 & 0 & 0 & 6720 & 0 & 60910 & 0 & 94080 & 0 \\
144 & 0 & 0 & 0 & -4368 & 0 & -38640 & 0 & -148701 & 0 & -80640 \\
0 & 0 & 0 & 1330 & 0 & 16800 & 0 & 94080 & 0 & 164000 & 0 \\
0 & 0 & -144 & 0 & -2835 & 0 & -24192 & 0 & -80640 & 0 & -\frac{31452624}{691}
\end{pmatrix}$$

Fig. 5.2: The "normalised" algebraic period polynomial of Δ over $\mathbb{Q}(\sqrt{-11})$.

In analogy with the classical case (see Section 7.1 [Man73]), we expect the denominator of the first (or last) coefficient to give the modulus of a congruence. In this case we see a congruence modulo 691, and a 691 in the denominator.

This scaling gives the normalised algebraic period polynomial r_Δ. To detect a congruence with the Eisenstein series E_{12}, we need the *integrally* scaled polynomials R_Δ and $R_{E_{12}}$. We obviously have $R_{E_{12}} = r_{E_{12}} = X^{10}\overline{X}^{10} - Y^{10}\overline{Y}^{10}$, and $R_\Delta = 691 r_\Delta$.

From the above representation of r_Δ, it is clear every coefficient of R_Δ except the first and last is divisible by 691. So the reduction of R_Δ modulo 691 is

$$R_\Delta \equiv 377X^{10}\overline{X}^{10} - 377Y^{10}\overline{Y}^{10} \equiv 377R_{E_{12}} \pmod{691}.$$

Thus (although we already knew this), we have

$$\Delta \equiv E_{12} \pmod{691}.$$

Remark 5.4.3. The representation of our Bianchi period polynomials as square matrices has no particular significance; the polynomial is a vector in the space $W_{10,10}(K)^+$ and has no reasonable interpretation as a matrix that we know of. It is an aesthetic decision, as this representation fits more easily on the page, and its symmetries reflect some obvious symmetries of the polynomial under the action of $\mathrm{PSL}_2(\mathcal{O})$. For example, the matrix $S = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ acts on polynomials by $P(X, Y, \overline{X}, \overline{Y}) \cdot S = P(-Y, X, -\overline{Y}, \overline{X})$, which flips the matrix along the horizontal and vertical lines of symmetry and negates entries (i, j) with $i + j$ odd. Since

$r_\Delta \cdot (1-J) = 0$, where $J = \left(\begin{smallmatrix} -1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, all such entries are 0.

Remark 5.4.4. It may be worth noting that it appears to be possible to detect the prime 691 using only the "diagonal" coefficients of the period polynomial, i.e. those corresponding to the monomials $X^{10-i}Y^i\overline{X}^{10-i}\overline{Y}^i$, for $0 \leq i \leq 10$. Per Remark 5.4.1, these correspond to critical values of the L-function of Δ, and come from Magma unscaled as

$$\left(1, -\tfrac{7082750}{1965789}, \tfrac{4892971}{1497744}, -\tfrac{21044405}{15726312}, \tfrac{156857}{499248}, 0, -\tfrac{156857}{499248}, \tfrac{21044405}{15726312}, -\tfrac{4892971}{1497744}, \tfrac{7082750}{1965789}, -1\right).$$

Taking gcds as before reproduces the previous scaling, again putting a 691 in the denominator of the first and last coefficients.

5.4.7 *Congruence between the genuine cusp forms and E_{12}*

In the same manner as the base-change congruence, our goal is to compute the normalised period polynomial r_{F_1} and use it to detect a congruence with E_{12}. As the Hecke eigenvalues of F_1 live in the quadratic field $L = \mathbb{Q}(\sqrt{81829})$, any algebraic eigen-polynomial P realising the Hecke eigenvalue system of F_1 will live in $W_{10,10}(L')^+$, where L' is the compositum LK, a field of degree 4. We compute one such polynomial P and as before write

$$P = \sum_{0\leq i,j\leq 10} c_{i,j} X^{10-i}Y^i\overline{X}^{10-j}\overline{Y}^j.$$

Again, we scale P so that its leading coefficient is 1. In our next step, we would like to take the gcd of all the middle coefficients, however, since these coefficients lie in the number field L' whose class number[7] is not 1, we have to consider the gcd[8] of

[7] The field L' has class number 116.

[8] Here the notion of gcd for fractional ideals is the straightforward generalisation of the same for rational numbers used in Section 5.4.6.

the fractional ideals generated by the middle coefficients:

$$\mathcal{D} = \frac{\gcd\big(\{\langle \mathrm{numerator}(c_{i,j})\rangle \mid (i,j) \neq (0,0),(10,10)\}\big)}{\mathrm{lcm}\big(\{\langle \mathrm{denominator}(c_{i,j})\rangle \mid (i,j) \neq (0,0),(10,10)\}\big)}.$$

We find that the fractional ideal $\mathcal{D}$ is principal, generated by the element

$$D = (245047560419778865 \cdot T^3 + 4914499503886859704 67 \cdot T)/15099638400 \in L',$$

where T is a root of the polynomial $x^4 + 3725x^2 + 3448449$ so that $\mathbb{Q}(T) \simeq L'$. Now multiplying our period polynomial P by $1/D$, we obtain the normalised algebraic period polynomial r_{F_1}, whose middle coefficients all lie in the ring of integers of L' and generate ideals that are all pairwise coprime.

The key observation now is that the leading coefficient of our final polynomial is $1/D$, which has norm

$$N_{L'/\mathbb{Q}}(1/D) = \frac{2^{28} \cdot 3^4 \cdot 5^8 \cdot 7^4 \cdot 11^8}{173^2}.$$

This indicates the existence of a congruence modulo 173. To detect such a congruence at the level of period polynomials, we need to scale r_{F_1} such that *all* its terms are integral and coprime. A quick calculation shows that

$$D = \alpha/8131200,$$

where $\langle\alpha\rangle$ is the unique prime of L' of norm 173^2, and so $R_{F_1} = \alpha r_{F_1}$. As we saw for the previous congruence, all the middle coefficients of R_{F_1} are divisible by α, so modulo $\langle\alpha\rangle$, R_{F_1} is in the space generated by $R_{E_{12}}$. Writing $\mathfrak{p}_{173}$ for the unique prime of norm 173 in L, another quick calculation shows that $\langle\alpha\rangle \cap \mathcal{O}_L = \mathfrak{p}_{173}$, i.e. $\mathfrak{p}_{173}$ is inert in L'/L. In particular, we have

$$R_{F_1} \equiv 27 R_{E_{12}} \pmod{\langle\alpha\rangle}.$$

From this we conclude that

$$F_1 \equiv E_{12} \pmod{\mathfrak{p}_{173}}.$$

Remark 5.4.5. As in Remark 5.4.4, we also examine the unscaled diagonal entries of F_1:

$$\left(1, -\frac{3287}{924}, \frac{42039}{13552}, -\frac{173}{154}, \frac{173}{968}, 0, -\frac{173}{968}, \frac{173}{154}, -\frac{42039}{13552}, \frac{3287}{924}, -1\right)$$

This choice of scaling is rather non-intrinsic, although it is surprising that there is *any* choice of scaling such that all the diagonal coefficients are rational, as they *a priori* should be algebraic integers in L (per Hida's result in (5.9)). Scaling them as before (to clear gcds into the denominators of the first and last entries), we get

$$\left(\frac{40656}{173}, -836, 729, -264, 42, 0, -42, 264, -729, 836, -\frac{40656}{173}\right).$$

Although this choice of scaling does *not* reproduce the scaling obtained from considering the whole polynomial, we do see the factor of 173 reappearing the denominator of the first and last terms. We don't know if scaling via the diagonal coefficients should always be sufficient to detect a congruence, or if one can expect there to be a choice of scaling that makes the diagonal rational in all cases.

The interested reader can find all the unnormalised algebraic period polynomials (including those corresponding to F_1 and F_2, which we cannot display here due to their size) in the associated GitHub repository, in the file `Q11_periodPols.m`.

5.4.8 *Congruence between the genuine cusp forms and* Δ

Finally, we look to detect congruences between the genuine cusp forms F_1, F_2 and the lifted form Δ. It is a well-known result of Hida that such congruences are controlled by the size of a certain "congruence module" which, by a result of Urban [Urb95] for Bianchi cusp forms, is captured by the algebraic part of the value of the adjoint

L-function at $s = 1$ (which is obtained by factoring out suitable pair of complex periods).

Further, it is well-known that the adjoint L-function at $s = 1$ is closely related to the Petersson norm. In turn, a result of Haberland [Hab83], see also [KZ84, PP13], tells us that one can compute the Petersson norm of a classical cusp form via its canonical period polynomial. This motivates us to try to detect congruences between our genuine cusp forms and Δ using our Bianchi period polynomials.

Let $f \in S_{k+2}(\Gamma)$ be a Bianchi cusp form and let ω_f be its $V_{k,k}(\mathbb{C})$-valued harmonic differential form on the associated hyperbolic 3-fold X_Γ. Recall that the Petersson norm of f is given by

$$(f, f) = \int_{X_\Gamma} \omega_f \wedge {}^*\omega_f$$

where ${}^*\omega_f$ is the harmonic 2-form given by the Hodge-star of ω_f.

We believe that a suitable analogue of Haberland's formula which expresses the Petersson norm of f in terms of its canonical period polynomials ought to exist, however our preliminary efforts failed to establish such a formula. Instead, we have developed a computational approach that we believe captures the algebraic part of the Petersson norm. We explain this approach now.

Any Bianchi cusp eigenform $f \in S_{k+2}(\Gamma)$ gives rise to a class in $H^1(\Gamma, V_{k,k}(\mathbb{C}))$ as we have discussed earlier, but also to a class[9] in $H^2(\Gamma, V_{k,k}(\mathbb{C}))$. As previously discussed, an algebraic period polynomial can be associated to the class of f in H^1. For the second cohomology class, we use the fact that there is a Hecke-module isomorphism

$$H^2(\Gamma, V_{k,k}(\mathbb{C})) \simeq V_{k,k}(\mathbb{C})/\mathbf{V}(\mathbb{C}).$$

(see Section 4.3). Using computer programs that we developed, we can compute an algebraic vector v_f in $V_{k,k}(K(f))$ such that the class of v_f in $V_{k,k}(K(f))/\mathbf{V}(K(f))$ realises the Hecke eigenvalue system of f. Thus, the class of v_f corresponds to the

[9] The description of the $V_{k,k}(\mathbb{C})$-valued harmonic differential 2-form associated to f can be found in Section 3 of [Hid94].

cohomology class associated to f in H^2. We then take the pairing $\langle r_f, v_f \rangle$. We note that the value of $\langle r_f, v_f \rangle$ does not change if we replace v_F with $v_F + \mathbf{v}$ for any $\mathbf{v} \in \mathbf{V}$, since $\mathbf{V}$ and $W_{k,k}$ are orthogonal under $\langle \cdot, \cdot \rangle$. Our expectation is that the algebraic quantity $\langle r_f, v_f \rangle$ captures the algebraic part of the Petersson norm of f.

We now apply this strategy to the forms in $W_{10,10}(\mathrm{PGL}(\mathcal{O}_{11}))$, looking first at Δ. We have already written down a normalised algebraic period polynomial corresponding to Δ in Section 5.4.6. We also compute an algebraic vector v_Δ in $V_{10,10}(K)$ whose class in $V_{10,10}(K)/\mathbf{V}(K)$ realises the Hecke eigenvalue system of Δ:

$$v_\Delta = -358X^{10}\overline{X}^8\overline{Y}^2 + 3080X^{10}\overline{X}^6\overline{Y}^4 + 22253X^{10}\overline{X}^4\overline{Y}^6.$$

It is not clear exactly how to normalise v_Δ. The most natural way would be to match what we do for period polynomials, making all coefficients (except possibly the first and last) integral. But the quotient by $\mathbf{V}(K)$ throws away a lot of information, and it is plausible that terms with denominators don't show up in v_Δ.

For lack of a better convention, we have stuck to eliminating denominators and making the gcd of all (except the first and last) coefficients 1. We then compute

$$\langle r_\Delta, v_\Delta \rangle = \frac{7^2 \cdot 13 \cdot 43}{2}. \tag{5.10}$$

Since 7 is less than the weight 10, we do not expect to see a congruence at that prime. There is no cusp form in $W_{10,10}$ congruent to Δ modulo 13, but we *do* observe that

$$\Delta \equiv F_1 \pmod{\mathfrak{p}_{43}},$$

for the first few primes $\mathfrak{p}$ of K, with $\mathfrak{p}_{43}$ the unique prime of L of norm 43. To show this congruence holds for all primes of K, we need to check the integrally scaled period polynomials R_Δ and R_{F_1} live in the same space modulo $\mathfrak{q}_{43}$, the ideal of norm 43^2 in L', which satisfies $\mathfrak{q}_{43} \cap \mathcal{O}_L = \mathfrak{p}_{43}$. As before this just means that $\mathfrak{p}_{43}$ is inert in L'/L.

We already have the scaled period polynomials, and reducing them both modulo $\mathfrak{q}_{43}$, we find

$$R_\Delta \equiv 5R_{F_1} \pmod{\mathfrak{q}_{43}},$$

and so

$$\Delta \equiv F_1 \pmod{\mathfrak{p}_{43}},$$

as indicated by the copy of 43 appearing in the pairing in (5.10).

The fact that 43 appears both in the pairing and as the modulus of a congruence here *could* be a coincidence, so we compute the same pairing, this time with F_1. Again computing, we have

$$N(\langle r_{F_1}, v_{F_1}\rangle) = 2^2 \cdot 5^4 \cdot 7^4 \cdot 13^4 \cdot 43^4 \cdot 173^2$$

We again discard the primes less than the weight 10. We again see 43 appearing, suggesting this construction does capture congruence moduli. We also note that

$$F_1 \equiv F_2 \pmod{\mathfrak{p}},$$

where $\mathfrak{p} \in \{\mathfrak{p}_{11}, \mathfrak{p}_{43}, \mathfrak{p}_{173}\}$, since F_1 and F_2 are L-conjugate, and each of $\{11, 43, 173\}$ ramifies in L. As they come from the structure of the coefficient field L, these congruences are much less interesting the others.

Meanwhile, the reappearance of 13 in the product is, again, mysterious, and we cannot account for it. We note that it does *not* come from a congruence with a torsion class mod 13, as there is no 13-torsion in $H^2(\Gamma_{11}, V_{10,10}(\mathcal{O}_{11}))$, per Table 13 of [§11].

6. COMPUTATIONS WITH GALOIS REPRESENTATIONS

This chapter mainly concerns the practical implementation of the theory of Selmer groups given in Chapter 3. To this end, we will make frequent reference to code contained in Appendix A, and commands from the computer algebra package Magma [BCP97]. We will cover the details that makes the algorithm effective, and give examples that illustrate the methods. The aim of this chapter is to give as complete an account as we can as to how to associate Selmer groups and period information to a given Galois representation over $K = \mathbb{Q}$ or an imaginary quadratic field, as well as gather data around the distribution of ranks of Selmer groups, and speculate on possible connections between the two.

6.1 Computing Selmer groups

6.1.1 Class field theory

We collect some results from class field theory we will need. Good references are [Lan14], [Coh12]. For this section, L is a general number field.

Definition 6.1.1. A **modulus** in L is a formal product

$$\mathfrak{m} = \prod_{\mathfrak{q}} \mathfrak{q}^{e_{\mathfrak{q}}}$$

over all primes $\mathfrak{q}$ of L, both finite and infinite, with all exponents satisfying $e_{\mathfrak{q}} \geq 0$.

Additionally, we require $e_{\mathfrak{q}} = 0$ for all but finitely many $\mathfrak{q}$, and, at infinite primes $\mathfrak{q}$,

$$\begin{cases} e_{\mathfrak{q}} = 0 & \text{when } \mathfrak{q} \text{ is complex,} \\ e_{\mathfrak{q}} \leq 1 & \text{when } \mathfrak{q} \text{ is real.} \end{cases}$$

A modulus can therefore be written as a (formal) product $\mathfrak{m}_0\mathfrak{m}_\infty$, where $\mathfrak{m}_0$ is an ideal of $\mathcal{O}_L$ and $\mathfrak{m}_\infty$ is a formal product of real places. By convention, we take the empty product (i.e. when all $e_{\mathfrak{q}} = 0$) to be the trivial ideal $\mathcal{O}_L$.

Definition 6.1.2. The **ray class group associated to** $\mathfrak{m}$ is the group $\mathrm{Cl}_{\mathfrak{m}}(L)$, defined as the quotient of

$$I_{\mathfrak{m}}(L) = \{\mathfrak{n} \trianglelefteq \mathcal{O}_L \mid \gcd(\mathfrak{n}, \mathfrak{m}) = 1\},$$

the group of fractional ideals of L coprime to $\mathfrak{m}$, by

$$P_{\mathfrak{m}}(L) = \{\mathfrak{n} \trianglelefteq \mathcal{O}_L \mid \gcd(\mathfrak{n}, \mathfrak{m}) = 1,\ \mathfrak{n} = \alpha\mathcal{O}_L,\ \alpha \equiv 1 \pmod{{}^*\mathfrak{m}}\},$$

the subgroup of principal fractional ideals of L coprime to $\mathfrak{m}$, generated by an element α congruent to 1 (mod $\mathfrak{m}_0$) and positive at all embeddings of $\mathfrak{m}_\infty$. This final condition is denoted by $\alpha \equiv 1 \pmod{{}^*\mathfrak{m}}$ in the definition.

It is a classical theorem of class field theory (see Chapter 3 of [Coh12]) that $\mathrm{Cl}_{\mathfrak{m}}(L)$ is a finite abelian group, and that there is an abelian extension $L(\mathfrak{m})/L$ such that

1. The extension $L(\mathfrak{m})/L$ is unramified for $\mathfrak{q} \nmid \mathfrak{m}$;
2. There is an isomorphism of Galois groups

$$\mathrm{Cl}_{\mathfrak{m}}(L) \simeq \mathrm{Gal}(L(\mathfrak{m})/L)$$

   sending a prime $\mathfrak{q}$ to $\mathrm{Frob}_{\mathfrak{q}} \in \mathrm{Gal}(L(\mathfrak{m})/L)$. This is the *Artin map.*

The modulus $\mathfrak{m} = 1$ gives rise to $\mathrm{Cl}_{\mathfrak{m}}(L) = \mathrm{Cl}(L)$, the usual class group of L, and the ray class field is H/L, the Hilbert class field of L.

Finally, we can bound the ramification of a prime $\mathfrak{p}$ of L using Proposition 3.3.22 of [Coh12], which gives the largest possible exponent we need to take a given prime $\mathfrak{p}$ to in order to get the maximal abelian p-extension of L, unramified outside of p. For our fields L, we find this maximal exponent $\mathcal{E}(\mathfrak{p})$ to be

$$\mathcal{E}(\mathfrak{p}) = \left\lceil 2e + \frac{e}{p-1} + 1 \right\rceil, \tag{6.1}$$

where e is the ramification index of p in L, p the characteristic of ρ.

6.1.2 Galois representations, again

Let $\rho\colon G_K \to \mathrm{GL}(V)$ be a continuous 2-dimensional irreducible mod p Galois representation. By making a choice of basis of V we have that $\mathrm{GL}(V) \leq \mathrm{GL}_2(\mathbb{F}_q)$ for some $q = p^r$. This choice is arbitrary, but for computational purposes it doesn't matter which choice we make. Recall from Definition 2.2.2 the *splitting field* of ρ:

$$L = \overline{K}^{\ker(\rho)},$$

satisfying $\mathrm{Gal}(L/K) \simeq \mathrm{im}(\rho) \simeq G_K/G_L$. We use the extension L/K to compute with ρ. To do this, we use `Magma`'s `IrreducibleModules` command. This finds all irreducible representations of $\mathrm{Gal}(L/K)$ over a finite field $\mathbb{F}$. We can further filter by dimension, leaving us with a list of irreducible 2-dimensional representations of $\mathrm{Gal}(L/K)$ over $\mathbb{F}$, which can then be disambiguated by their traces of Frobenius. We also require the representation to be faithful, to ensure $\mathrm{im}(\rho) \simeq \mathrm{Gal}(L/K)$, rather than just a subgroup.

The code to do this is contained in `get_rep.m`, in Section A.2.2, and was originally written by Aurel Page, adapted somewhat to our specific situation.

6.1.3 Computing the relaxed Selmer group

Let $\mathcal{L}$ be a Selmer system for ρ. Recall from Theorem 3.3.2 and Section 3.3.3, lines in $\mathrm{Sel}_{\mathcal{L}}(\rho)$ are in bijection with extensions M/L satisfying

1. The extension M/L is Galois, with $\mathrm{Gal}(M/L) \simeq V$ (as an additive group);
2. The extension M/K is Galois;
3. The action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(M/L)$ is via ρ;
4. The ramification of primes in M/L is controlled by the local conditions in $\mathcal{L}$.

Using this theorem, we can compute the rank of $\mathrm{Sel}_{\mathcal{L}}(\rho)$ exactly. In practice, we first compute the relaxed Selmer group, which we recall puts the unramified condition on all places of of L not over the characteristic p, and no condition over p. In terms of extensions as in point (4) above, M/L must be unramified primes of L away from p only. For simplicity of exposition, we will assume $V = \mathbb{F}_p \oplus \mathbb{F}_p$, but the same works for $V = \mathbb{F}_q \oplus \mathbb{F}_q$, if one interprets $\mathbb{F}_q = \mathbb{F}_{p^r} \simeq \mathbb{F}_p^r$ as $\mathbb{F}_p$-vector spaces.

From Equation 6.1, we know we can find a modulus $\mathfrak{m}$ such that $L(\mathfrak{m})$ is the maximal abelian extension of L unramified away from p. We can then find a subfield $A = A_{\mathfrak{m}} \subset L(\mathfrak{m})$ such that A/L is also unramified away from p, and $\mathrm{Gal}(A/L) \simeq \mathbb{F}_p^k$. Thus, any extension M/L satisfying conditions (1-4) is a subfield of A.

Practically, this is all accomplished using the Magma commands `RayClassGroup` to find $\mathrm{Cl}_{\mathfrak{m}}(L)$ and `AbelianExtension` to find $L(\mathfrak{m})$. We then find A as the fixed field of the largest elementary p-subgroup of $\mathrm{Cl}_{\mathfrak{m}}(L)$. To satisfy condition (2), we use Magma's `NormalSubfields` command, which filters out only those M/L that are also normal M/K; by passing in the additional parameter `Quot:=[p,p]`, we can further filter for only those M/L with Galois group isomorphic to $V \simeq \mathbb{F}_p \oplus \mathbb{F}_p$. As subfields of an abelian extension, all the M/L are automatically Galois, satisfying condition (2). Finally, during the computation of A, Magma automatically gives the action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(A/L)$, so we can compare with the action of $\mathrm{Gal}(L/K)$ on V we have already computed and check the two are conjugate.

Often, the action of $\mathrm{Gal}(L/K)$ on an extension satisfying conditions (1,2,4) will not be via ρ, and in fact, will not even necessarily have image equal to $\mathrm{im}(\rho)$. We will see this in numerous cases, including, for example, Example 6.1.3.

Example 6.1.3. Let $K = K_3 = \mathbb{Q}(\sqrt{-3})$, with $\omega = \omega_3 = \frac{1+\sqrt{-3}}{2}$. Let

$$E : y^2 + xy + \omega y = x^3 + (\omega + 1)x^2$$

be the elliptic curve with LMFDB label `4219.1-a1`. Its associated 2-torsion representation is cut out by the field L/K, defined by the polynomial

$$\begin{aligned} x^6 + (-2\omega + 8)x^5 + (-17\omega + 43)x^4 + (-52\omega + 128)x^3 + \\ (-85\omega + 206)x^2 + (-74\omega + 168)x - 24\omega + 46, \end{aligned} \tag{6.2}$$

so we have $\mathrm{Gal}(L/K) \simeq S_3 \simeq \mathrm{GL}_2(\mathbb{F}_2)$. Since S_3 is generated by an element τ of order 2 and an element σ of order 3 such that $\tau\sigma\tau = \sigma^2$, we can describe the action of the whole Galois group in terms of two such elements. Let α be a root of (6.2). Then we can define τ and σ to be

$$\begin{aligned} \tau \colon \alpha \mapsto \frac{1}{9813}((-2998\omega + 2102)\alpha^5 + (-17800\omega + 7865)\alpha^4 + \\ (-89194\omega + 29504)\alpha^3 + (-214598\omega + 75421)\alpha^2 + \\ (-259530\omega + 81459)\alpha + (-115570\omega + 18401)) \end{aligned}$$

$$\begin{aligned} \sigma \colon \alpha \mapsto \frac{1}{19626}((1201\omega + 1567)\alpha^5 + (-1085\omega + 15130)\alpha^4 + \\ (-16391\omega + 72871)\alpha^3 + (-62857\omega + 171614)\alpha^2 + \\ (-122412\omega + 176442)\alpha + (-52664\omega + 14824)). \end{aligned}$$

Per Section 3.4.3, we have

$$H^1(\mathrm{GL}_2(\mathbb{F}_2), V) = H^2(\mathrm{GL}_2(\mathbb{F}_2), V) = 0,$$

so lines in the Selmer group correspond exactly to extensions M/L satisfying conditions (1-4) above.

In $K/\mathbb{Q}$, the prime 2 is inert, and in L/K it splits into 3 primes: $2\mathcal{O}_L = \mathfrak{p}_1^2\mathfrak{p}_2^2\mathfrak{p}_3^2$. By Equation 6.1, we can take the modulus $\mathfrak{m} = (\mathfrak{p}_1\mathfrak{p}_2\mathfrak{p}_3)^7$. Magma gives

$$\mathrm{Cl}_{\mathfrak{m}}(L) \simeq (\mathbb{Z}/2\mathbb{Z})^2 \oplus (\mathbb{Z}/4)^2 \oplus (\mathbb{Z}/8\mathbb{Z})^5.$$

Thus, we have

$$\mathrm{Gal}(A_{\mathfrak{m}}/L) \simeq (\mathbb{Z}/2\mathbb{Z})^9.$$

When doing class field theory with Magma, we keep all extensions of L in the data type `FldAb`. Computationally, it is much quicker to leave everything as this type; converting to number fields and performing arithmetic there tends to be much slower.

For example, finding the subfields $A/M/L$ that are normal over K is computationally very taxing when A is converted to a number field. As a `FldAb`, the action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(A/L)$ is already computed, and so we can use standard techniques to find the subgroups of $\mathrm{Gal}(A/L)$ that correspond to such fields. In particular, to compute $\mathrm{Sel}_{\mathrm{rel}}(\rho)$, we only need to find the subgroups that are also submodules with respect to the $\mathrm{Gal}(L/K)$ action, which Magma does with ease.

The action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(A_{\mathfrak{m}}/L)$ is given by

$$\tau \mapsto \begin{pmatrix} 0&1&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&1&0\\ 1&1&1&0&0&0&0&0&1\\ 1&0&1&1&0&0&0&1&1\\ 0&0&0&0&1&0&0&1&1\\ 1&1&1&1&1&1&1&1&1\\ 0&1&1&0&0&0&1&1&0\\ 0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&1&0 \end{pmatrix}, \quad \sigma \mapsto \begin{pmatrix} 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&1&1\\ 1&1&0&1&1&0&0&0&1\\ 0&1&1&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&1\\ 0&0&0&0&0&1&1&1&0\\ 1&1&0&0&1&1&0&0&0\\ 0&0&0&0&0&0&0&1&1\\ 0&0&0&0&0&0&0&1&0 \end{pmatrix}.$$

Using Magma's `Submodules` command, we find ten submodules of $\mathrm{Gal}(A_{\mathfrak{m}}/L)$ isomorphic to $(\mathbb{Z}/2\mathbb{Z})^7$, which map, under the Galois correspondence, to extensions

M/L with Galois group V. These ten extensions M/L break down into classes based on the image of the action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(M/L)$: seven have image S_3, two have image C_2, and one has image C_1.

All of the seven extensions with S_3-action have that action conjugate to ρ, so from Lemma 3.4.1, we find that

$$\mathrm{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{rel}}(\rho)) = 3.$$

This computation is performed with `NormalSubfields_K.m`, in Section A.2.4.

6.1.4 Computing the nearly-ordinary Selmer group

To compute $\mathrm{Sel}_{\mathrm{NO}}(\rho)$, we will assume we have already computed $\mathrm{Sel}_{\mathrm{rel}}(\rho)$. Recall that ρ is nearly-ordinary at a place w of K if $\rho\,|_{D_w}$ fixes a 1-dimensional subspace $\ell \leq V$. The definition of $\mathrm{Sel}_{\mathrm{NO}}(\rho)$ requires that ρ be nearly-ordinary at all places over p.

Recall also that the corresponding extra condition on extensions M/L satisfying conditions (1-3) in Theorem 3.3.2 that marks out elements of $\mathrm{Sel}_{\mathrm{NO}}(\rho)$ is that $I_{\mathfrak{q}}(M/L) \leq \ell$ for all places $\mathfrak{q} \mid \mathfrak{p}$. Here we are implicitly using the identification of $\mathrm{Gal}(M/L)$ with V (as an additive group). In fact, we will see that having $I_{\mathfrak{q}}(M/L) \leq \ell$ for one place $\mathfrak{q}$ of L is sufficient.

Proposition 6.1.4. Let ρ be nearly-ordinary at $\mathfrak{p}$. Then, for two choices $\mathfrak{q}$ and $\mathfrak{q}'$ of places of L above $\mathfrak{p}$, if $I_{\mathfrak{q}}(M/L)$ is in the line fixed by $D_{\mathfrak{p}}^{\mathfrak{q}}(L/K)$, then $I_{\mathfrak{q}'}(M/L)$ is in the line fixed by $D_{\mathfrak{p}}^{\mathfrak{q}'}(L/K)$.

Proof. The group $\mathrm{Gal}(L/K)$ acts transitively on the primes of L, so there is some $\sigma \in \mathrm{Gal}(L/K)$ such that $\mathfrak{q} = \sigma\mathfrak{q}'$. Then we have

$$D_{\mathfrak{p}}^{\mathfrak{q}'}(L/K) = \sigma^{-1} D_{\mathfrak{p}}^{\mathfrak{q}}(L/K)\sigma.$$

Call ℓ the line fixed by $D_{\mathfrak{p}}^{\mathfrak{q}}(L/K)$. A quick calculation shows that $D_{\mathfrak{p}}^{\mathfrak{q}'}(L/K)$ fixes $\ell \cdot \rho(\sigma)$. The result follows from the fact that $I_{\mathfrak{q}}(M/L) \cdot \rho(\sigma) = I_{\mathfrak{q}'}(M/L)$, which we now prove.

Take $\tau \in I_{\mathfrak{q}}(M/L)$ and $x \in \mathcal{O}_{M/L}$. Recall from Section 3.3.2 that $\rho(\sigma)$ acts on $\mathrm{Gal}(M/L)$ by conjugation by some extension $\tilde{\sigma}$ of σ to $\mathrm{Gal}(M/K)$. Write $y = \tilde{\sigma}(x)$; then we have

$$\begin{aligned}(\tau \cdot \rho(\sigma))(x) - x &= (\tilde{\sigma}^{-1}\tau\tilde{\sigma})(x) - x \\ &= (\tilde{\sigma}^{-1}\tau)(y) - \tilde{\sigma}^{-1}(y) \\ &= \tilde{\sigma}^{-1}(y+q) - \tilde{\sigma}^{-1}(y) \\ &= \tilde{\sigma}^{-1}(q).\end{aligned}$$

where $q \in \tilde{\mathfrak{q}}$, a place of M extending $\mathfrak{q}$. The action of $\tilde{\sigma}^{-1}$ takes elements of $\tilde{\mathfrak{q}}$, to elements of $\tilde{\mathfrak{q}'}$, a place of M extending $\mathfrak{q}'$.

We want to argue that $\tau \cdot \rho(\sigma) \in I_{\mathfrak{q}'}(M/L)$, but this group *a priori* depends on the choice of a place extending $\mathfrak{q}'$, with the groups coming from these choices all conjugate in $\mathrm{Gal}(M/L)$. But $\mathrm{Gal}(M/L)$ is abelian, so this choice isn't really a choice (conjugacy classes have size 1), and so $\tau \cdot \rho(\sigma) \in I_{\mathfrak{q}'}(M/L)$ as required. □

Testing whether a representation is nearly-ordinary at a place $\mathfrak{p}$ is easy: the decomposition group can be computed using (2.2), and we can test directly whether the fixed points of $\rho\,|_{D_{\mathfrak{p}}}$ form a line in V.

Similarly, we can test whether an inertia group $I_{\mathfrak{q}}(M/L)$ lies in ℓ directly. To compute the inertia subgroup of M/L at $\mathfrak{q}$, we find the maximal unramified subextension M' of M. This is exactly $M^{I_{\mathfrak{q}}(L/K)}$, and it can be found by intersecting M with the maximal abelian extension of L unramified at every place not over p, *and* unramified at $\mathfrak{q}$.

This field is the ray class field attached to the modulus $\mathfrak{m}/\mathfrak{q}^{e_{\mathfrak{q}}}$—the same as the modulus defining A, but with every copy of $\mathfrak{q}$ excised. Then we have

$$M^{I_{\mathfrak{q}}(M/L)} = M \cap L(\mathfrak{m}/\mathfrak{q}^{e_{\mathfrak{q}}}).$$

Meanwhile, we can find the extension M^ℓ by first computing a generator of the line ℓ, then using the isomorphism $V \simeq \mathrm{Gal}(M/L)$. We have $I_\mathfrak{q}(M/L) \leq \ell$ if and only if $M^\ell \leq M^{I_\mathfrak{q}(L/K)}$, which can be checked directly in Magma.

This computation is performed with `NO_selmer_lines.m`, in Section A.2.5.

Example 6.1.5. We continue with the representation ρ from Example 6.1.3. We pick the prime $\mathfrak{p}_1 = (2, \beta)$ to compute decomposition and inertia groups, where

$$\begin{aligned}\beta =& \frac{1}{19626}((\omega - 17198)\alpha^5 + (-10010\omega - 11105795)\alpha^4 + \\ &(16867\omega - 65046950)\alpha^3 + (25334\omega - 56392243)\alpha^2 \\ &+ (24786\omega + 136309662)\alpha + (-9146\omega + 44855554)).\end{aligned}$$

A quick calculation in Magma shows that

$$D_2(L/K) = \langle \tau\sigma \rangle \simeq C_2,$$

We identify $\mathrm{Gal}(L/K)$ with $\mathrm{GL}_2(\mathbb{F}_2)$ by

$$\rho \colon \tau \mapsto \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \; \sigma \mapsto \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix},$$

so that $\rho(\tau\sigma) = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, and $\rho\mid_{D_2(L/K)}$ fixes the line $\ell = \langle (1,1) \rangle$. Of the seven extensions M/L from Example 6.1.3, three satisfy $I_{\mathfrak{p}_1}(M/L) \leq \ell$, so again by Lemma 3.4.1, we have that

$$\mathrm{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{NO}}(\rho)) = 2.$$

If we had chosen to compute our decomposition and inertia groups with $\mathfrak{p}_2 = \mathfrak{p}_1\sigma$, we would have found that

$$D_2(L/K) = \langle \tau\sigma^2 \rangle \simeq C_2,$$

and therefore $\rho\mid_{D_2(L/K)}$ instead fixes the line $\ell' = \langle (0,1) \rangle = \ell\rho(\sigma)$. Then the same three fields again have $I_{\mathfrak{p}_2}(M/L) \leq \ell'$ by the compatibility of the $\mathrm{Gal}(L/K)$ action everywhere.

6.1.4.1 Computing the unramified Selmer group

This Selmer group is easy to compute if one is also computing the nearly-ordinary group, since that computation requires finding $M^{I_{\mathfrak{q}}(M/L)}$ for all $\mathfrak{q}$ over $\mathfrak{p}$. The unramified Selmer group is then found by counting all those M for which

$$[M^{I_{\mathfrak{q}}(M/L)} : L] = [M : L]$$

i.e. those for which the inertia group is trivial.

This calculation is also performed by `NO_selmer_lines.m`.

Example 6.1.6. Again we continue with the representation ρ from Examples 6.1.3, 6.1.5. Since we have already computed the $M^{I_{\mathfrak{q}}(M/L)}$, we only need to check the list. Of the seven, only one has degree 4 over L, corresponding to $M^{I_{\mathfrak{q}}(M/L)} = M$, so we have

$$\operatorname{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{unr}}(\rho)) = 1.$$

6.1.5 The p-Selmer group of an elliptic curve

For an elliptic curve E over a number field K, there is a notion of the *p-Selmer group* of E. It is defined similarly to our Selmer groups (see Section X.4 of [Sil09], for example), and is used in computing the rank of E. We should not, in general, expect the ranks of our mod p Selmer groups for representations coming from an elliptic curve E to match the rank of the p-Selmer group of E, or indeed the rank of E itself.

The relaxed Selmer group generally over-estimates the rank of E, and the unramified Selmer group generally under-estimates it. The nearly-ordinary Selmer group, sitting between the two, has the best chance of giving the rank of E, but there exist many examples where it does not.

The elliptic curve E appearing in Examples 6.1.3-6.1.6 happens to have rank 2, matching this nearly-ordinary rank of its mod 2 representation.

6.2 Computing period data

Having explored the Selmer side, we now want to associate period data to a Galois representation. To do this, we compute its Serre conductor, Serre weights and its character, and find the (conjecturally) associated eigenvalue system(s) in cohomology.

Example 6.2.1. Let $K = K_7 = \mathbb{Q}(\omega)$, where $\omega = \frac{1+\sqrt{-7}}{2}$. Let $\rho\colon G_K \to \mathrm{GL}_2(\mathbb{F}_5)$ be the representation that cuts out the field extension $L = K(\alpha)$, where α is a root of

$$x^8-4x^7+8x^6-10x^5+(-6\omega+18)x^4+(12\omega-24)x^3+(-12\omega+21)x^2+(6\omega-10)x-\omega+2.$$

We choose this extension because it has a relatively small Serre conductor, meaning we can compute the cohomology groups where its (conjecturally) associated eigenvalue system is found with ease.

The field L has Galois group $\mathrm{Gal}(L/K) \simeq D_4 = \langle \sigma, \tau \rangle$, where $\sigma^4 = \tau^2 = (\sigma\tau)^2 = 1$. We make the explicit identifications:

$$\begin{aligned}\sigma\colon \alpha \to & \frac{1}{1562}((-216\omega-732)\alpha^7 + (756\omega+2562)\alpha^6 + (-1208\omega-4788)\alpha^5 \\ & + (1130\omega+5565)\alpha^4 + (3004\omega-13944)\alpha^3 + (-5258\omega+16632)\alpha^2 \\ & + (5134\omega-12453)\alpha - 1671\omega + 4360)\end{aligned}$$

and

$$\begin{aligned}\tau\colon \alpha \to & \frac{1}{1562}((516\omega-1896)\alpha^7 + (-1806\omega+6636)\alpha^6 + (3580\omega-11992)\alpha^5 \\ & + (-4435\omega+13390)\alpha^4 + (16948\omega-21880)\alpha^3 + (-21890\omega+22748)\alpha^2 \\ & + (19149\omega-13076)\alpha - 6031\omega + 3816).\end{aligned}$$

We also realise the map ρ by

$$\rho\colon \sigma \mapsto \begin{pmatrix} 0 & 1 \\ 4 & 0 \end{pmatrix}, \quad \tau \mapsto \begin{pmatrix} 0 & 4 \\ 4 & 0 \end{pmatrix}.$$

To find the associated eigenvalue system in cohomology, we need to compute the invariants from Section 2.3.

Serre conductor: Recall that the Serre conductor is the ideal

$$\mathfrak{n}(\rho) = \prod_{\substack{\mathfrak{p}\text{ prime} \\ \mathfrak{p}\nmid p}} \mathfrak{p}^{e_{\mathfrak{p}}(\rho)},$$

where $e_{\mathfrak{p}}(\rho)$ is an exponent defined using the higher ramification groups of $\operatorname{im}(\rho)$, and p is the characteristic of ρ. In particular, $e_{\mathfrak{p}}(\rho) = 0$ when ρ is unramified at $\mathfrak{p}$, which happens exactly when L/K is. The discriminant of $\mathcal{O}_L$ is $\mathfrak{p}_5^4\mathfrak{p}_{29}^4$, where $\mathfrak{p}_5 = (5)$ has norm 25, and $\mathfrak{p}_{29} = (1-4\omega)$ has norm 29. Since ρ has characteristic 5, we only need to know $e_{\mathfrak{p}_{29}}(\rho)$.

Choosing a prime $\mathfrak{q}_{29}$ of L over $\mathfrak{p}_{29}$, we compute the higher ramification groups

$$G_{\mathfrak{p}_{29},0} = \langle \sigma^3\tau \rangle \simeq C_2, \quad G_{\mathfrak{p}_{29},1} = C_1.$$

The image of $G_{\mathfrak{p}_{29},0}$ under ρ is $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 4 \end{smallmatrix}\right)$, which fixes the 1-dimensional subspace $\langle (1,0) \rangle$ of $\mathbb{F}_5 \oplus \mathbb{F}_5$ pointwise. So we compute $e_{\mathfrak{p}_{29}}(\rho) = 1$, and so

$$\mathfrak{n} = \mathfrak{n}(\rho) = (1-4\omega).$$

All of this computation is performed by `conductor.m`, in Section A.1.2.

Character: The character χ of the representation ρ is a (mod 5) Dirichlet character of the group $(\mathcal{O}_K/\mathfrak{n})^\times$, such that

$$\chi(\pi) = \det \rho(\mathrm{Frob}_{(\pi)})$$

for prime elements $\pi \in \mathcal{O}_K$ not dividing $\mathfrak{n}p$. Recall we can make this identification between ideals an elements since K has class number 1. There are 28 characters to check against in this case, and we find that χ is the unique quadratic character of $(\mathcal{O}_K/\mathfrak{n})^\times$, reduced modulo 5.

This computation is performed by `character.m`, in Section A.1.1.

Traces of Frobenius: Having computed the Frobenius elements to find the character χ, we also note their traces here.

$\mathfrak{p}$	$(1+2\omega)$	$(9-2\omega)$	$(7+2\omega)$	$(7-6\omega)$	(13)	$(1-10\omega)$
$N(\mathfrak{p})$	11	71	71	79	169	191
$\mathrm{trace}(\mathrm{Frob}_\mathfrak{p})$	3	2	3	3	2	2

Tab. 6.1: Traces of Frobenius of ρ from Example 6.2.1.

For all other primes $\mathfrak{p}$ of norm ≤ 200, we have $\mathrm{trace}(\mathrm{Frob}_\mathfrak{p}) = 0$.

This computation is performed by the function `ComputeTraces`, in `get_rep.m`, Section A.2.2.

Weights: Since $p = 5$ is inert in K, we compute cohomology groups for the weight modules $V_{k,l}^{s,t,\chi}(\mathbb{F}_{25})$. Here we realise $\mathbb{F}_{25}$ as $\mathbb{F}_5(\mu)$, where $\mu^2 + 3\mu + 4 = 0$.

With the level and character in hand, we can compute the spaces $H^1(\Gamma_0(\mathfrak{n}), V_{k,l}^{s,t,\chi}(\mathbb{F}_{25}))$ for all $k, l, s, t \in \{0, \ldots, 4\}$, and find these traces as an eigenvalue system for (k, l, s, t) equal to

1. $(1, 1, 0, 0)$
2. $(1, 1, 2, 2)$
3. $(1, 1, 4, 4)$
4. $(2, 2, 1, 4)$
5. $(2, 2, 4, 1)$.

Period polynomials: Each occurrence of the eigenvalue system $a_{\mathfrak{p}} = \operatorname{trace}(\operatorname{Frob}_{\mathfrak{p}})$ is multiplicity 1, meaning its associated period polynomial can be determined, up to scaling by $\mathbb{F}_{25}$. For the weights (1-3), we find the period polynomial

$$r_\rho = X\overline{X} + Y\overline{Y}.$$

For the weights (4, 5) we find the period polynomial

$$r_\rho = XY\overline{X}^2 + X^2\overline{XY} + \mu Y^2\overline{XY} + \mu XY\overline{Y}^2.$$

As in Section 5.4.6, we can represent these period polynomials as matrices, which makes some of their symmetries more obvious:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & \mu \\ 0 & \mu & 0 \end{pmatrix}.$$

These period polynomials are computed using the code in Chapter B.2.

6.3 Interesting examples

The phenomena surrounding relaxed, nearly-ordinary and unramified Selmer groups of mod p Galois representations is relatively unexplored. As such, there are many examples displaying interesting behaviour for which little theory exists to account.

6.3.1 Different nearly-ordinary ranks

The elements we choose to include in the nearly-ordinary Selmer group depend on which line fixed by the decomposition group at $\mathfrak{p}$ we choose. In the case of only one fixed line, the choice is made for us, but there are some representations with trivial decomposition group, meaning $\rho\mid_{D_{\mathfrak{p}}}$ fixes all possible lines.

Example 6.3.1. Let $K = K_{11} = \mathbb{Q}(\sqrt{-11})$, with $\omega = \frac{1+\sqrt{-11}}{2}$. The elliptic curve

$$E : y^2 + \omega xy + y = x^3 + (-\omega + 1)x^2 - \omega x$$

has LMFDB label `17713.1-a1`. Its mod 2 representation ρ cuts out the splitting field L, defined by the polynomial

$$x^6 - 3x^5 + (-6\omega + 22)x^4 + (12\omega - 39)x^3 + (-6\omega + 79)x^2 - 60x + 16,$$

so $\mathrm{im}(\rho) \simeq \mathrm{GL}_2(\mathbb{F}_2)$. We note that L has no real places. There is only one prime of K over 2, which is $2\mathcal{O}_K$, and it factors as a product of 6 primes, $2\mathcal{O}_L = \mathfrak{q}_1 \dots \mathfrak{q}_6$ in L. Per Equation (6.1), we can compute the maximal 2-extension A/L unramified outside $\mathfrak{q} \mid 2\mathcal{O}_K$ with the modulus $\mathfrak{m} = (2\mathcal{O}_K)^4$. From Magma, we get

$$\mathrm{Gal}(A/L) \simeq (\mathbb{Z}/2Z)^{13}.$$

Further, we find fifteen extensions M/L with action of $\mathrm{Gal}(L/K)$ on $\mathrm{Gal}(M/L)$ via ρ. Thus, we have

$$\mathrm{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{rel}}(\rho)) = 4.$$

Next, we look at the nearly-ordinary Selmer groups. The decomposition group is the trivial group $C_1 \leq \mathrm{GL}_2(\mathbb{F}_2)$, so $\rho\mid_{D_{2\mathcal{O}_K}}$ fixes every line of $V \simeq \mathbb{F}_2 \oplus \mathbb{F}_2$. These lines are

$$\ell_1 = (1,0), \quad \ell_2 = (1,1), \quad \ell_3 = (0,1).$$

Let $\mathfrak{q} = (2, \beta)$ be an ideal of L over $2\mathcal{O}_K$, with

$$\begin{aligned}\beta = &\frac{1}{1676}((-1750\omega - 377)\alpha^5 + (-15306\omega + 15629)\alpha^4 + \\ &(26666\omega - 17762)\alpha^3 + (-55590\omega + 97855)\alpha^2 + \\ &(43528\omega - 84305)\alpha - 12260\omega + 26708).\end{aligned}$$

Then we count the extensions M/L such that $I_{\mathfrak{q}_1}(M/L) \leq \ell_i$. For ℓ_1 we count 3, for

ℓ_2 we count 1, and for ℓ_3 we count 7. This gives

$$\mathrm{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{NO}(\ell_1)}(\rho)) = 2, \quad \mathrm{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{NO}(\ell_2)}(\rho)) = 1, \quad \mathrm{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{NO}(\ell_3)}(\rho)) = 3.$$

This example illustrates that the rank of the nearly-ordinary Selmer group need not be constant across different lines. We also note that only one of the M/L is unramified at primes $\mathfrak{q}$ over $2\mathcal{O}_K$, and so

$$\mathrm{rank}_{\mathbb{F}_p}(\mathrm{Sel}_{\mathrm{unr}}(\rho)) = 1.$$

Finally, we also note that the rank of E is 2.

6.3.2 Ramified and unramified quotient

In this subsection we are motivated by comments made in Section 5.1 of [EPW05], where the relationship between the Greenberg and Bloch-Kato Selmer groups of a p-adic representation coming from a classical modular form is discussed. The Greenberg Selmer group is defined in Section 4.1 of that paper, and can be applied to representations that are

- nearly-ordinary in the exactly analogous way as outlined in Section 3.3.3.2;
- unramified at p on the quotient V/ℓ, where ℓ is the space fixed by $\rho\mid_{D_p}$.

It is noted that the Greenberg Selmer group always contains the Bloch-Kato Selmer group with corank at most 1.

In the search for a local condition for mod p representations that shares the (conjectural) relationship between periods and ranks as outlined in Section 1.2, we investigate whether a such a property holds here. That is, for a nearly-ordinary mod p representation ρ with fixed line ℓ providing a quotient character $\rho^* : G_K \to V/\ell$ unramified at $\mathfrak{p}$, is it the case that the rank of $\mathrm{Sel}_{\mathrm{NO},\ell}(\rho)$ is non-zero when a period of the mod p cohomology class is 0?

We analyse this situation for mod 3 representations of $G_{\mathbb{Q}}$, coming from elliptic curves, with image $\mathrm{SD}_{16} \leq \mathrm{GL}_2(\mathbb{F}_3)$. The group SD_{16} has order 16, and has presentation

$$\mathrm{SD}_{16} \simeq \langle \sigma, \tau \mid \sigma^2 = \tau^8 = 1, \sigma^{-1}\tau\sigma = \tau^3 \rangle.$$

This can be realised as a subgroup of $\mathrm{GL}_2(\mathbb{F}_3)$ by, for example,

$$\sigma \mapsto \begin{pmatrix} 2 & 0 \\ 2 & 1 \end{pmatrix}, \quad \tau \mapsto \begin{pmatrix} 0 & 2 \\ 2 & 2 \end{pmatrix}.$$

Up to conductor 500,000, we find 186 such representations coming from elliptic curves that are nearly-ordinary at 3. In each case, we found that $\rho\mid_{D_3}$ fixed two lines ℓ_1, ℓ_2, with V/ℓ_1 unramified at 3, and V/ℓ_2 ramified. So for each representation, we compute an unramified nearly-ordinary rank and a ramified one. We note the difference between the unramified nearly-ordinary (hereafter, "UNO") rank and the unramified rank; the latter captures those elements of the Selmer group that are unramified *everywhere*, not just on the quotient V/ℓ_1. We summarise the data gathered in the table below.

rank(E)	Relaxed	Unramified N.O.	Ramified N.O.	Unramified
0.6701	1.5876	0.9587	0.7525	0.2371

Tab. 6.2: Average ranks associated to SD_{16} mod 3 representations coming from elliptic curves.

We note that the UNO rank is larger than the ramified nearly-ordinary rank on average.

We now test the naive generalisation of the comments of [EPW05] to our mod 3 representations. We compute $\mathrm{rank}_{\mathbb{F}_3}(\mathrm{Sel}_{\mathrm{NO},\ell}(\rho_{E,3}))$ in the case of the unramified quotient, and match this against period information. The speculation states that this rank should overestimate the "true" rank by at most 1.

In order to verify this speculation with the UNO Selmer group, we must do the following.

1. Compute the period $\mathcal{P}(\overline{f}_E)$;
2. Compute the rank of the UNO Selmer group.
3. Compare the two. If the period is
   (a) zero, then the rank should be at least 1.
   (b) not zero, then the rank should be at most 1 (as the hypothesised "true rank" should be 0, and the UNO Selmer group should overestimate this quantity by at most 1)

For an elliptic curve E with rank ≥ 1, the value $\mathcal{P}(f_E)$ is predicted to be 0 by the Birch and Swinnerton-Dyer conjecture; for all curves in the LMFDB this has been verified to hold, so we can take it as read that $\mathcal{P}(\overline{f}_E) = 0$.

When E has rank 0, the value $\mathcal{P}(f_E)$ is non-zero for the same reason: all the curves we consider have the Birch and Swinnerton-Dyer conjecture verified. However, the value $\mathcal{P}(\overline{f}_E)$ can be 0, which can happen when the integrally scaled f_E has a 3 in its numerator.

In order to comport with the speculation that the UNO rank should overestimate the "true" rank by at most 1, in the case of a rank 0 curve, we should see a UNO rank of at most 1, assuming $\mathcal{P}(\overline{f}_E)$ does not vanish.

In practice, we find several examples of elliptic curves with rank 0 with UNO rank 2, seemingly violating this principle. One such curve is

$$E\colon y^2 + xy + y = x^3 + x^2 - 1048096x + 551582926,$$

with LMFDB label `24649.a1`. However, this curve (conjecturally) has non-trivial Shaferavich-Tate group; in this case, the value $|\text{Ш}(E)| = 9$ is predicted by the (conjectural) formula for $L(E,1)$, and is likely contributing a factor of 3 to the value $\mathcal{P}(f_E)$. The spaces $H^1(\Gamma_0(24649), F)$ for $F = \mathbb{C}, \mathbb{F}_3$ are outside our computational capability, so we cannot verify these speculations.

For the 186 representations considered, we find 6 curves where the difference between the UNO rank and the rank of E is larger than 1, seeming to violate this speculation. However, in each case, a factor of 3 appears in the numerator of the ratio $\frac{L(E,1)}{\Omega_E}$.

Four of the curves (`24649.a1`, `264992.dm1`, `288800.ba1` and `453152.bq1`) have a factor of 3 coming from the order of $Ш(E)$; the other two curves (`94178.bb1` and `153760.d1`) have a factor of 3 coming from a Tamagawa number. In all cases, we expect then that the value $\mathcal{P}(\overline{f}_E)$ vanishes, meaning the "true" rank should be at least 1, and thus the UNO rank 2 only overestimates by 1, as predicted.

6.3.3 Elliptic curves with large rank

Let E be the elliptic curve `111061427.a1`, given by

$$E\colon y^2 + xy = x^3 - 245x + 1366.$$

This curve has rank 5, and is nearly-ordinary at 2. Computing its Selmer groups, we find

$$\mathrm{rank}(\mathrm{Sel}_{\mathrm{rel}}(\rho_{E,2})) = 5,$$

$$\mathrm{rank}(\mathrm{Sel}_{\mathrm{NO}}(\rho_{E,2})) = 5,$$

$$\mathrm{rank}(\mathrm{Sel}_{\mathrm{unr}}(\rho_{E,2})) = 4.$$

For the hypothetical "correct" Selmer system for mod 2 representations, one should expect the rank to be close to the rank of E. Here the two match, and it is notable that most of the rank comes from the unramified portion of the Selmer group, with only one extra dimension truly coming from the nearly-ordinary condition.

6.4 Statistics on nearly-ordinary ranks

For each of the five $K = K_d$ we consider, and the rationals $K = \mathbb{Q}$, we present statistics on the ranks of nearly-ordinary Selmer groups coming from elliptic curves over K. To collect the data, we started with the LMFDB elliptic curve databases over K. From this, we kept every elliptic curve satisfying the following properties:

1. The conductor $\mathfrak{n}(E)$ is not divisible by any prime of K over 2;
2. The image of $\rho_{E,2}$ is $\mathrm{GL}_2(\mathbb{F}_2)$
3. The representation $\rho_{E,2}$ is nearly-ordinary at all primes of K over 2.

Next, using field isomorphism testing, we sorted the representations into isomorphism classes. As a first pass, we computed the traces of Frobenius for each curve for primes less than 1000, to create probable isomorphism classes. After this, we ran over each class and isomorphism tested each member to find exact isomorphism classes. In practice, no probable isomorphism class ever split.

Recall from Section 6.3.1, a mod p Selmer group can have different nearly-ordinary ranks associated to each fixed line; for a small proportion of representations ($\sim 5\%$), the decomposition group over 2 vanished, fixing all lines. We count each of these as a separate Selmer group.

Lastly, of the five fields, 2 splits only in $\mathbb{Q}(\sqrt{-7})$, and so it is possible to get up to 9 different nearly-ordinary Selmer groups, which occurs when the decomposition groups of both primes over 2 vanish. In practice, we did not see this occur.

We present the data gathered in the following table.

Field	$\mathbb{Q}$	$\mathbb{Q}(\sqrt{-3})$	$\mathbb{Q}(\sqrt{-1})$	$\mathbb{Q}(\sqrt{-7})$	$\mathbb{Q}(\sqrt{-2})$	$\mathbb{Q}(\sqrt{-11})$
2	–	inert	ramified	split	ramified	inert
Max conductor	500000	150000	100000	50000	50000	50000
Reps	75445	3342	1530	441	1200	1711
Proportion N.O.	0.5103	0.7101	0.6895	0.1723	0.7067	0.7078
N.O. reps	38497	2373	1055	76	848	1211
N.O. Selmer groups	48927	2573	1105	94	894	1307
Vanishing	0	131	26	4	38	137
Proportion van.	0	0.0509	0.0235	0.0426	0.0425	0.105
Mean relaxed	2.493	3.054	3.028	4.118	3.037	3.059
Mean N.O.	1.517	1.383	1.877	1.947	1.488	1.284
Mean unramified	0.409	0.0737	0.0569	0.0790	0.0637	0.0751

Tab. 6.3: Various statistics for nearly-ordinary representations over $\mathbb{Q}$ and five imaginary quadratic fields K.

6.4.1 Features of the data

Here we comment on some notable features of the collected data.

6.4.1.1 Proportion of nearly-ordinary representations

Firstly, it appears that a representation over $\mathbb{Q}$ is nearly-ordinary around half of the time. Meanwhile, it appears that the probability of a representation satisfying (1-3) above being nearly-ordinary is around 70% when there is one prime in K above 2, and somewhere around 17% when there are two. It is “harder” for a representation to be nearly-ordinary over the field where 2 splits, since it must have decomposition groups that fix a line for two primes, rather than one.

If the probability of a decomposition group fixing a line over K_7 is also around 70%, as it appears to be for the fields where 2 is inert or ramified, and the probabilities for the two primes over 2 in K_7 are independent, one would expect that the proportion of nearly-ordinary representations there would be around $0.49 = 0.7^2$. But the observed proportion is much lower. We conclude that some aspect of this naive interpretation is clearly not correct, though we do not know which.

6.4.1.2 Proportion of vanishing nearly-ordinary ranks

We note that the nearly-ordinary Selmer group seems to never vanish over $\mathbb{Q}$; the same is *not* true for these K, although the proportion still remains low, around 1-5%.

6.4.1.3 Average ranks

The nearly-ordinary rank across the five number fields sits somewhere between 1 and 2. It is largest for K_7, although this may just be a result of the much smaller data set there. The relaxed rank and unramified ranks are also larger for this field, possibly for the same reason.

Over $\mathbb{Q}$, the rank is very close to 1.5. We note that the nearly-ordinary rank seems to never vanish, meaning it is overestimating the rank of the underlying elliptic curves of rank 0 by at least 1. The current consensus seems to be that the average rank of elliptic curves (ordered by conductor) should be $\frac{1}{2}$; it is possible, then, that the nearly-ordinary rank is overestimating the elliptic curve rank by 1 on average.

In general, the relaxed rank seems to sit very close to 3, and the unramified rank is small, around 0.06. In particular, we saw no Selmer group with unramified rank above 1 across the five fields. The largest relaxed rank we see is 5, from a few representations over K_7, for example the 2-torsion representation of the elliptic curve `1269.2-a1`. No representation over another field reaches this relaxed rank in this data range, although examples such the one in Section 6.3.3 indicate this may be just because of the ranks of the underlying curves not being large enough.

As a possible alternative explanation for the higher ranks seen over K_7, we make the following observation: the size of the ray class field of a modulus $\mathfrak{m}$ tends to increase as its number of factors does (with multiplicity). Since there are two primes over 2 in K_7, the modulus we define in Section 6.1.1 has more factors, and so ought to lead to a larger number of possible extensions corresponding to lines in the Selmer group.

Appendices

Appendix A

CODE FOR GALOIS REPRESENTATIONS

A.1 Invariants

A.1.1 `character.m`

```
// assuming we already have the representation rho, the frobenius data
// from aurel's code, its conductor cond,
// and the characteristic p of the representation

RepChar:=function(rho,frob,cond,p)

  ZK:=Order(cond);

  DG1:=DirichletGroup(cond);
  DG2:=DirichletGroup(p*ZK);

  E1:=Elements(DG1);
  E2:=Elements(DG2);

  // we find the order of the determinant character
  // when the rep lives in some non-prime finite field
  frob_order:=1;
  frob_dets:=[u[3] : u in frob];
```

```
frob_test:=[u^1 : u in frob_dets];
while not &and [u eq 1 : u in frob_test] do
  frob_order+:=1;
  frob_test:=[u^frob_order : u in frob_test];
end while;

possible_chars:=[];
// now we run over all the characters in both, looking
// for d1 and d2 such that their product gives the determinant
// character. then d1 is the character of the rep as predicted by Serre
for d1 in E1 do
  for d2 in E2 do
    if LCM(Order(d1),Order(d2)) eq frob_order then
      test:=[d1(u[1])*d2(u[1]) : u in frob];
      kk:=Rationals();
        for u in test do
          par:=Parent(u);
          if Type(par) ne RngInt then
            kk:=Compositum(kk,Parent(u));
          end if;
        end for;
      Zkk:=MaximalOrder(kk);
      pp:=Factorization(p*Zkk)[1,1];
      ff,down:=ResidueClassField(pp);
      if [down(u) : u in test] eq frob_dets then
      Append(~possible_chars,<d1,Index(E1,d1)>);
      end if;
    end if;
  end for;
end for;

if #possible_chars eq 1 then
  return possible_chars[1][1], possible_chars[1][2];
  elif #possible_chars eq 0 then
    print "no character found";
    return -1;
  else
    print "multiple possible characters found";
    print "more determinants of frobenius needed to disambiguate";
    return possible_chars;
  end if;
end function;
```

A.1.2 `conductor.m`

```
// given a representation rep, returned by Aurel's galreps function,
// a prime P of ZLL (the ring of integers of L/K, entry 3 in rep), and an
// index i, returns the higher ramification group (lower numbered)
// of P of index i
HigherRamGroup:=function(rho,P,i)

  A:=Domain(rho[1]);

  ram_group:=[];
  B:=Basis(rho[3]);

  Q:=Factorization(Parent(1*rho[3])!P)[1,1];
  CC:=rho[4];

  for g in A do
    keep:=true;
    for j in [1..#B] do
      if not (rho[2](g)(B[j])-B[j])*CC[j] in Q^(i+1) then
        keep:=false;
        break j;
      end if;
    end for;
    if keep then
      Append(~ram_group,g);
    end if;
  end for;
  return sub<A|ram_group>;
end function;


// given a representation rho and a prime P of ZLL,
// returns the exponent of P in the Serre conductor of rho
SerreExponent:=function(rho,P)

  ram_groups:=[HigherRamGroup(rho,P,0)];
  i:=1;
  while #ram_groups[#ram_groups] ne 1 do
    Append(~ram_groups,HigherRamGroup(rho,P,i));
    i+:=1;
  end while;

  ind:=0;
  for R in ram_groups do
    Kers:=[];
    for r in R do
      m:=rho[1](r);
      S:=Kernel(m-m^0);
      Append(~Kers,S);
    end for;
    d:=Dimension(&meet Kers); // dimension of the space they all fix
    ind+:=#R/#ram_groups[1]*(2-d);
  end for;

  return Integers()!ind;
```

```
end function;


SerreConductor:=function(rho)

  K:=BaseField(NumberField(rho[3]));
  ZK:=MaximalOrder(K);
  p:=Characteristic(Codomain(rho[1]));
  primes_over_char:=[u[1] : u in Factorization(p*ZK)];

  ZLLdisc:=Discriminant(rho[3]);

  cond:=1*ZK;

  for P in Factorization(ZLLdisc) do
    if &and [Valuation(P[1],u) eq 0 : u in primes_over_char] then
      // excludes primes over the characteristic
      cond*:=P[1]^SerreExponent(rho,P[1]);
    end if;
  end for;

  return cond,Factorization(cond);
end function;
```

A.2 Selmer groups

A.2.1 `CFT_utility.m`

```
// QOL function, gives the version of Q(sqrt(-d))
// such that the basis of K is a Z-basis of ZK
QuadFld:=function(d)
  _<x>:=PolynomialRing(Integers());
  if d in [1,2] then
    return NumberField(x^2+d);
  elif d in [3,7,11] then
    return NumberField(x^2-x-Integers()!((-d-1)/4));
  else
    return "d = " cat Sprint(d) cat " currently not supported";
  end if;
end function;


// returns the map mm: R2 -> \{ideals of ZL\} that cuts out the p subext
// of RayClassField(m). so RayClassField(mm) should give a group (Z/p)^n
MaxlPExt:=function(R,m,p)
  h:=hom< R-> R | [p*R.i : i in [1..Ngens(R)]]>;
  Q,mq:=quo<R | Image(h) >;

  qm:=Inverse(mq);
  mm:=qm * m;

  return mm;
end function;


// takes the a list of actions and returns the stats on what groups the
// actions encode on the vector space (Z/p)\^{}2
ProfileNSF:=function(ACT,p)
  group\_stats:=[];
  group\_list:=[];
  for u in ACT do
    H:=MatrixGroup<2,GF(p)|u>;
    name:=GroupName(H);
    names:=[g[1] : g in group\_stats];
    if not name in names then
```

```
      Append(\~{}group\_stats,[*name,1*]);
    else
      group\_stats[Index(names,name)][2]+:=1;
    end if;
    Append(\~{}group\_list,name);
  end for;
  return group\_stats,group\_list;
end function;
```

A.2.2 `get_rep.m`

Remark A.2.1. We note that this code was adapted from code originally written by Aurel Page.

```
function isfaithful(rho)
  G := Domain(rho);
  MR := Codomain(rho);
  for g in G do
    if g ne G!1 and rho(g) eq MR!1 then
      return false;
    end if;
  end for;
  return true;
end function;

function ffss2dreps(G,q) //faithful semisimple
  F<w>:=GF(q);
  Lirr := IrreducibleModules(G,F);
  if IsAbelian(G) then
    Ldim1 := [M : M in Lirr | Dimension(M) eq 1];
    Ldim2 := [Representation(M) : M in Lirr | Dimension(M) eq 2];
    Lreps := [Representation(DirectSum(Ldim1[i],Ldim1[j])) :
      j in [i..#Ldim1], i in [1..#Ldim1]];
    Lreps cat:= Ldim2;
  else
    Lreps := [Representation(M) : M in Lirr | Dimension(M) eq 2];
  end if;
    Lreps := [rho : rho in Lreps | isfaithful(rho)];
  return Lreps;
end function;

function galreps(F,K,q)
  G,_,f := AutomorphismGroup(F,K);
  if #G ne AbsoluteDegree(F) div AbsoluteDegree(K) then
    print "not Galois, early abort";
    return [];
  end if;
  L := ffss2dreps(G,q);
  FF:=RelativeField(K,F);
  ZFF:=MaximalOrder(FF);
  CC:=[];
  PP:=PseudoBasis(Module(ZFF));
  for u in PP do
    tt,gg:=IsPrincipal(u[1]);
    Append(~CC,gg);
  end for;
  return [<rho,f,ZFF,CC> : rho in L];
end function;

function isfrob(basisZL,aut,N,proj)
  for x in basisZL do
    if proj(x)^N ne proj(aut(x)) then
      return false;
    end if;
  end for;
  return true;
end function;
```

```
function frobenius(ZL,f,pr) // f: G -> Aut_K(L)
  G := Domain(f);
  a,b := TwoElement(pr);
  facto := Factorization(ideal<ZL|ZL!a,ZL!b>);
  PR := facto[1,1];
  if facto[1,2] ne 1 then
    return 0; //ramified!
  end if;
  N := Norm(pr);
  Q,proj := ResidueClassField(PR);
  basis:=Basis(ZL);
  for g in G do  //could be improved
    aut := f(g);
    if isfrob(basis,aut,N,proj) then
      return g;
    end if;
  end for;
  print "should not be reached!";
  return 0;
end function;

function frobmatrix(ZL,galrep,pr)
  rho := galrep[1];
  f := galrep[2];
  g := frobenius(ZL,f,pr);
  return rho(g);
end function;

detect_hnf:=function(J,M)
  N:=Norm(J);
  a:=M[1,1];  d:=M[1,2];
  b:=M[2,1];  c:=M[2,2];
  if (d eq 0) and (N eq a*c) and (b in [0..a-1]) then
    return 1;
  else
    return 0;
  end if;
end function;

HNF_basis:=function(J)
  N:=Norm(J);
  M:=BasisMatrix(J);
  Mt:=Matrix(Integers(),2,2,[M[1,2],M[1,1],M[2,2],M[2,1]]);
  HN:=HermiteForm(Mt);
  H:=Matrix(Integers(),2,2,[HN[2,2],HN[2,1],HN[1,2],HN[1,1]]);
  c:=H[2,2];   b:=H[2,1];    a:=H[1,1];
  if c lt 0 then
    c:=-c;    b:=-b;
  end if;
  b:=(b mod a);
  assert 1 eq detect_hnf(J,Matrix(Integers(),2,2,[a,0,b,c]));
  return [Norm(J), b,c];
end function;

ComputeFrob:=function(ZL,K,rho, max)
  ZK:=MaximalOrder(K);
  PP:=[J : J in IdealsUpTo(max,K) | IsPrime(J) and GCD(J,Discriminant(ZL)*ZK) eq
  if Degree(K) ne 1 then
    HNF:=[HNF_basis(J): J in PP];
    ParallelSort(~HNF,~PP);
  end if;
  traces:=[* *];
  for i in [1..#PP] do
    J:=PP[i];
    t,g:=IsPrincipal(J);
    M:=frobmatrix(ZL,rho,J);
    Append(~traces,<g, Trace(M),Determinant(M)>);
  end for;
  return traces;
end function;
```

A.2.3 nearly_ordinary.m

```
// acts on an ideal by an automorphism
AutIdeal:=function(aut,ideal)
  gens:=Generators(ideal);
  R:=Ring(Parent(ideal));
  return ideal<R | [aut(g) : g in gens]>;
end function;


// computes decomp group
DecompGroup:=function(rep,P)
  G:=Domain(rep[1]);
  decomp:=[];
  gens:=SetToSequence(Generators(G));
  for g in gens do
    if AutIdeal(rep[2](g),P) eq P then
      Append(~decomp,g);
    end if;
  end for;
  return sub<G|decomp>;
end function;


// creates the projective line over F_p.
// returns lines (a,b) with a,b in GF(p) covering all such possible lines
// in projective space, and a function r that recognises a line (c,d) in
// its "canonical" form
ProjLineFp:=function(p)
  F:=GF(p);
  line:=[];
  for i in [0..p-1] do
    Append(~line,Vector([F!1,F!i]));
  end for;
  Append(~line,Vector([F!0,F!1]));
  r:=function(v)
    if v[1] eq 0 then
      return Vector([F!0,F!1]),1/F!v[2];
    else
      return Vector([1,v[2]/F!v[1]]),1/F!v[1];
    end if;
  end function;
  return line,r;
end function;


// computes the action of a matrix on the projective line PL
ProjActionV:=function(mat,PL,r)
  p:=Characteristic(CoefficientRing(mat));
  M:=ZeroMatrix(GF(p),p+1);
  for i in [1..p+1] do
    ind:=Index(PL,r(PL[i]*mat));
    M[i,ind]:=1;
  end for;
  return M;
end function;


// for rep a rep, D the decomposition group under consideration, PL
// and r coming from ProjActionV, computes the lines in F_p^2 fixed
// by all elements of D
FixedLines:=function(rep,D,PL,r)
  spaces:=[Kernel(ProjActionV(rep[1](D.i),PL,r)-1) : i in [0..Ngens(D)]];
  S:=&meet spaces;
  lines:=[];
  F:=CoefficientRing(PL[1]);
  p:=Characteristic(F);
  vz:=Vector([F!0 : i in [1..p+1]]);
  for i in [1..p+1] do
    v:=vz;
    v[i]:=1;
    if v in S then
      Append(~lines,PL[i]);
    end if;
  end for;
  return lines;
end function;


// ZL ring of integers of extension, P a prime ideal of ZL
// tells you if rho|D_P fixes any 1 dim'l spaces, and what the
// spaces are if it does
IsNearlyOrdinaryP:=function(rep,P)
  D:=DecompGroup(rep,P);
  p:=Characteristic(Codomain(rep[1]));
  PL,r:=ProjLineFp(p);
  lines:=FixedLines(rep,D,PL,r);
  if #lines ne 0 then
    return true,lines;
  else
    return false,[];
  end if;
end function;


// tells you if the mod p rep is nearly ordinary at all places over p
// does a bunch of redundant stuff, we only need to check one prime per
// ideal of K over p, since Galois action shuffles the fixed lines around
// also returns the fixed lines
IsNearlyOrdinary:=function(ZL,rep)
  p:=Characteristic(Codomain(rep[1]));
  fac:=[f[1] : f in Factorization(p*ZL)];
  nearly_ord:=true;
  ss:=0;
  for P in fac do
    t,ss:=IsNearlyOrdinaryP(rep,P);
    if not t then
      nearly_ord:=false;
      break P;
    end if;
  end for;
  return nearly_ord,ss;
end function;
```

A.2.4 `NormalSubfields_K.m`

Remark A.2.2. We note that this code is adapted from Magma's own code to compute normal subfields. Our main contribution has been to fix compatibility issues so that relative fields could be used, which took no original mathematical thought.

```
//turns a vector to a group element, pretty straightforward
// in particular it is compatible with the rest of the functions here
VecToGroup:=function(vec,grp)
  elt:=Id(grp);
  for i in [1..Ncols(vec)] do
    elt+:=Integers()!vec[i]*grp.i;
  end for;
  return elt;
end function;


CreateHom := function(G, GG, mp)
  m1 := Matrix(Integers(), [Eltseq(x[1]) : x in mp]);
  m1 := VerticalJoin(m1, DiagonalMatrix([ Order(G.x) : x in [1..Ngens(G)]]));
  _, t := EchelonForm(m1);
  m1 := Matrix(Integers(), [Eltseq(x[2]) : x in mp]);
  t := Matrix(Integers(), [ Eltseq(t[i])[1..#mp] : i in [1..Nrows(t)]]);
  m1 := t*m1;
  mp := [ GG!Eltseq(m1[i]) : i in [1..Ngens(G)]];
  h := hom<G -> GG | mp>;
  return h;
end function;



InducedMap_K:=function(r1,r2,h,Coprime)
  pp := NextPrime(100);
  li := [ ];
  lg := [ ];
  G := Domain(r1);
  H := Domain(r2);
  o := Ring(Codomain(r1));

  HEval:=function(h,I)
    gens:=Generators(I);
    R:=Order(I);
    h_gens:=[h(g) : g in Generators(I)];
    h_ideal:=ideal<R | h_gens>;
    return h_ideal;
  end function;

  repeat
    repeat
      pp := NextPrime(pp);
    until Gcd(pp, Coprime) eq 1;
    lp := Decomposition(o, pp);
    for i in lp do
      if Degree(i[1]) gt 1 and Norm(i[1]) gt 1000 then
      continue;
    end if;
    Append(~lg, i[1] @@ r1);
    Append(~li, HEval(h,i[1]) @@ r2);
    end for;
  until sub<G|lg> eq G;

  return CreateHom(G, H, [<lg[i], li[i]> : i in [1..#lg]]);
end function;


function convert_K(elt, Mk, M, mo)
  X := Domain(M);
  Z := Domain(Mk);
  phi := Mk(elt);
  aut := InducedMap_K(M,M, phi, mo);
  return Matrix([Eltseq(aut(X.i)) : i in [1..Ngens(X)]]);
end function;


CohomologyModule_K:=function(F,Sub,K)
```

```
  k := BaseField(F);
  g, _, p := AutomorphismGroup(k,K);

  if Sub cmpne false then
    g := Sub;
  end if;

  A, mo := NormGroup(F);
  mo := AbsoluteNorm(mo);
  AA := InvariantRepresentation(Domain(A));
  mAA := Coercion(AA, Domain(A));

  inv := AbelianInvariants(AA);
  mats := [ convert_K(g.i, p, mAA*A, mo) : i in [1..Ngens(g)]];

  C := CohomologyModule(g, inv, mats);
  Zm := RSpace(Integers(), Ngens(AA));
  mp := map<Zm -> AA | x :-> AA!Eltseq(x), y:-> Zm!Eltseq(y)>;

  return C, p, mAA*A, mp;
end function;


// given GalMats acting on Q and a subgroup S, find the action on
// the space Q/S
GalMatsQuo:=function(Q,S,GalMats)

  Mats:=[];
  Q1,down:=quo<Q|S>;
  up:=Inverse(down);
  p:=Exponent(Q);

  for gg in GalMats do
```

```
    N:=Ngens(Q1);
    vecs:=[down(VecToGroup(ChangeRing(Vector(ElementToSequence(
      up(Q1.i))),GF(p))*gg,Q)) : i in [1..N]];
    Append(~Mats,Matrix(GF(p),N,N,[ElementToSequence(v) : v in vecs]));
  end for;

  return Mats;
end function;



NormalSubfields_K:=function(A,Quot,K)
  All:=true;
  Over:=false;
  N := NormGroup(A);



  print "finding Aut(L/K)...";
  g, _, mg := AutomorphismGroup(BaseField(A),K); // this is Gal(L/K)
  print "creating cohomology module (using class group data)...";
  q1, q2, q3, q4 := CohomologyModule_K(A,false,K);

  // this part gives us the action of Gal(L/K) on the ray class group Q
  // this is stored in the GalMats list
  Q:=Domain(N);
  p:=Exponent(Q);
  G := Group(q1);
  a := [Q.i : i in [1..Ngens(Q)]];
  ChangeUniverse(~a, Domain(N));
  ChangeUniverse(~a, Domain(q3));
  b := [ChangeUniverse([ActionOnVector(q1, x@@q4, G.i)@q4 :
    x in a],Domain(N)) : i in [1..Ngens(G)]];
  GalMats:=[Matrix(GF(p),[ElementToSequence(e) : e in u]) : u in b];
  QInv:=Invariants(Q);
  print "finding submodules of chosen size...";
  GM:=GModule(g,MatrixAlgebra<GF(p),#QInv|GalMats>); // makes the Gal(L/K) modul
  LL:=Submodules(GM);

  // we want all those submodules which leave Quot as the quotient
  k:=#QInv - #Quot;

  keeps:=[];
  for u in LL do
    if Dimension(u) eq k then
      Append(~keeps,u);
    end if;
  end for;

  gens:=[];
  for u in keeps do
    Append(~gens,[VecToGroup(GM!u.i,Q) : i in [1..k]]);
  end for;

  subgroups:=[];
  for u in gens do
    Append(~subgroups,sub<Q|u>);
  end for;

  l := [AbelianSubfield(A, x:IsNormal, Over := Over) : x in subgroups];
  actions:=[GalMatsQuo(Q,s,GalMats) : s in subgroups];
  return l,actions,g,mg,gens;
end function;


// this fetches the action of G(L/K) on G(M/L). It returns
// 1) G(L/K) as a group of permutations
// 2) the action of these permutations on G(M/L) as matrices
// 3) and also as permutations
GetAction:=function(F,K)
  k := BaseField(F);
  g, _, p := AutomorphismGroup(k,K);

  A, mo := NormGroup(F);
  mo := AbsoluteNorm(mo);
  AA := InvariantRepresentation(Domain(A));
  mAA := Coercion(AA, Domain(A));

  Autos:=[];
  Mats:=[];

  for i in [1..#g] do
    Append(~Autos,InducedMap_K(mAA*A,mAA*A, p(g.i), mo));
    Append(~Mats,convert_K(g.i, p, mAA*A, mo));
  end for;

  return <g,Mats,Autos>;
end function;
```

A.2.5 NO_selmer_lines.m

```
// given a list of numbers [a_1,a_2,...,a_n], returns the list of all
// [u_1,u_2,...,u_n] such that 1 =< u_i =< a_i
LineCombos:=function(list)
  if #list eq 1 then
    return [[i] : i in [1..list[1]]];
  else
    combos:=[];
    prev:=$$(list[2..#list]);
    for u in prev do
      for i in [1..list[1]] do
        Append(~combos,[i] cat u);
      end for;
    end for;
    return combos;
  end if;
end function;


// takes the number N of lines, and the degree q of F_q
// and returns the rank
NumToRank:=function(N,q)
  qr:=(q-1)*N+1;
  b:=Floor(Log(q,qr));
  B:=Ceiling(Log(q,qr));
  if (q^b-1)/(q-1) eq N then
    return b;
  elif (q^B-1)/(q-1) eq N then
    return B;
  else
    print "bad number of lines " cat Sprint(N);
  return -N;
  end if;
end function;


// Takes a vector representating a line in V, and returns the
// subfield of M corresponding to M^V, i.e. the elements fixed by V
// currently only doing this for F_p reps, i.e. V = F_p + F_p
VecToSubfield:=function(M,vec)
  GM:=Domain(NormGroup(M)); // Gal(M/L)
  Z:=Integers();
  return AbelianSubfield(M,sub<GM|Z!(vec[1])*GM.1 + Z!(vec[2])*GM.2>);
end function;


// given two ideals I and J, returns the automorphisms that send
// I to J
IdentifyAutFromIdeals:=function(I,J,rep)
  auts:=[];
  G:=Domain(rep[1]);
  for i in [1..Ngens(G)] do
    if AutIdeal(rep[2](G.i),I) eq J then
      Append(~auts,G.i);
    end if;
  end for;
  return auts;
end function;


PrimeTranslate:=function(P,Q,rep)
  G:=Domain(rep[1]);
  gs:=[];
  for i in [1..Ngens(G)] do
    if AutIdeal(rep[2](G.i),P) eq Q then
      Append(~gs,G.i);
    end if;
  end for;
  return gs;
end function;


// given a line L fixed by a prime P, and another prime Q,
// finds an element g of the galois group such that the line g*L
// is fixed by Q. this is all wrt the action of rep. the line L is
// represented by its canonical generator from the nearly_ordinary code
// package. returns g and the vector v representing L in canonical form
LineTranslate:=function(L,P,Q,rep)

  G:=Domain(rep[1]);
  p:=Minimum(P);
  PL,r:=ProjLineFp(p);

  g:=-1;
  v:=-1;

  for i in [1..Ngens(G)] do
    if AutIdeal(rep[2](G.i),P) eq Q then
      g:=G.i;
      v:=r(L*rep[1](G.i));
      break i;
    end if;
  end for;
  return g,v;
end function;


// given one of our galois rep objects and the primes over its
// characteristic in L, returns the equivalence classes of its
// lines under the action of Gal(L/K). the order of the lines
// matches the order of the primes in the list given so if you
// pass rad (the radical) it should all line up nicely
FixedLineOrbits:=function(rep,rad)

  t,VP:=IsNearlyOrdinaryP(rep,rad[1]);
  orbits:=[];

  if t then
```

```
    for L in VP do
      new_orbit:=[];
      for Q in rad do
        g,v:=LineTranslate(L,rad[1],Q,rep);
        Append(~new_orbit,v);
      end for;
      Append(~orbits,new_orbit);
    end for;
  end if;

  return orbits;
end function;



// if rep1 and rep2 are conjugate and have the same domain, returns the
// element of G by which they are conjugate. if they aren't, returns "bad"
// this should never happen when used in REL_NO_UR.
ConjReps:=function(rep1,rep2)

  G:=Domain(rep1[1]);
  p:=Characteristic(Codomain(rep1[1]));

  act1:=[rep1[1](G.i) : i in [1..Ngens(G)]];
  act2:=[rep2[1](G.i) : i in [1..Ngens(G)]];

  V:=GL(2,p);

  for h in V do
    if [h^-1*u*h : u in act1] eq act2 then
      return h;
    end if;
  end for;
  print "something went wrong with conjugation";
  return "bad";
end function;



// returns the ranks of the relaxed, nearly-ordinary and
// unramified Selmer groups
REL_NO_UR:=function(S : Verbose:=false, Fast:=true, RamQuo:=false)

  PL,r:=ProjLineFp(S`p);

  // we collect the fixed lines over each prime over p
  fixed_lines:=[];
  for r in S`rad do
    tt,ll:=IsNearlyOrdinaryP(S`rep,r[1]);
    Append(~fixed_lines,ll);
  end for;

  unr_lines:=0;

  // this big list collates all the information about which
  // inertia groups lie
  // in which lines, which we then pass to a function that checks
  // all the combos
  AllFields_AllLines:=[];

  for i in [1..#S`NSF] do

    tr:=[Trace(u) : u in S`ACT[i]];
   M:=S`NSF[i];

    // we only keep the extensions with the right traces
// this means the representation of Gal(L/K) on Gal(M/L)
// is conjugate to S`rep
if S`traces eq tr then

  M_Inertia:=[];

  // collects all the fixed fields of inertia for comparison
  for r in S`rad do
    PP:=r[1];
    N, m, minf := NormGroup(S`A);
    Mur := RayClassField(m/PP^Valuation(m,PP), minf);
    MI:=Mur meet M;
    Append(~M_Inertia,MI);
  end for;

  // while we have the inertia groups, we do the checks for
  // the unramified Selmer group too
  is_unr:=true;
  for u in M_Inertia do
    if not Degree(u) eq Degree(M) then
      is_unr:=false;
      break u;
    end if;
  end for;

  if is_unr then
    unr_lines+:=1;
  end if;

  // we need to see how a given fixed line behaves inside
  // Gal(M/L) \simeq V so we figure out this isomorphism and
  // translate the line accordingly
  repM:=<Representation(GModule(S`AUT,S`ACT[i])),S`rep[2]>;
```

```
    h:=ConjReps(S`rep,repM);

    LineForP:=[];

    for j in [1..#S`rad] do

      ll_conj:=[r(u*h) : u in fixed_lines[j]];
      LineInInertia:=[];
      // if the subfield of M fixed by the line u is inside the
      // fixed field of inertia, the inertia group is in the line
      for u in ll_conj do
        Append(~LineInInertia,VecToSubfield(M,u) subset
          M_Inertia[j]);
      end for;
      Append(~LineForP,LineInInertia);
    end for;
    Append(~AllFields_AllLines,LineForP);
  end if;

end for;

// finally we go over AllFields_AllLines and collect the Selmer stats
// we check all possible combination of fixed lines over each prime
LineNums:=LineCombos([#u : u in AllFields_AllLines[1]]);

if Verbose then
  print "Number of fixed line combinations to check:", #LineNums;
end if;

NO_lines:=[];

for u in LineNums do
lines:=0;
```

```
  for R in AllFields_AllLines do
    contained:=[];
    for i in [1..#u] do
      Append(~contained,R[i][u[i]]);
    end for;
    if &and contained then
      lines+:=1;
    end if;
  end for;
  Append(~NO_lines,lines);
end for;

if Verbose then
  print "Fixed line inclusions for each " cat
    Sprint(GroupName(S`AUT)) cat " field:";
  AllFields_AllLines;
  "";
end if;

if RamQuo then
  // we compute the image of the inertia group of L/K at p
  // in the quotient V / l, where l is a fixed line
  chars:=[];
  for j in [1..#fixed_lines] do
    II:=InertiaGroup(S`rad[j][1]);
    new_list:=[];
    R:=RSpace(GF(S`p),2);
    for l in fixed_lines[j] do
      QQ,d:=quo<R|l>;
      Append(~new_list,[d(Inverse(d)(QQ.1)*S`rep[1](u))[1] :
        u in II]);
    end for;
    Append(~chars,new_list);
    end for;
    return [*NumToRank(#AllFields_AllLines,S`p),[NumToRank(u,S`p) :
      u in NO_lines],NumToRank(unr_lines,S`p)*],chars;
  else
    return [*NumToRank(#AllFields_AllLines,S`p),[NumToRank(u,S`p) :
      u in NO_lines],NumToRank(unr_lines,S`p)*];
  end if;
end function;
```

A.2.6 `selmer_ranks.m`

```
SEL:= recformat <
NSF                  : SeqEnum,
ACT                  : SeqEnum,
AUT                  : GrpPerm,
transfer             : Map,
group_list           : SeqEnum,
p                    : RngIntElt,
L                    : FldNum,
K                    : FldNum,
A                    : FldAb,
mm                   : Map,
rad                  : SeqEnum,
traces               : SeqEnum,
rep                  : Tup
>;

load "NormalSubfields_K.m";
load "CFT_utility.m";
load "nearly_ordinary.m";
load "NO_selmer_lines.m";
load "get_rep.m";
```

```
// for polynomial f over Q defining a number field L, with L/K the
// image of a  mod p Galois representation , returns the ranks of
// the relaxed , nearly-ordinary & unramified Selmer groups

// Ramification allows one to supply the ramifying primes of rho
// ahead of time, which can speed up MaximalOrder computations.
// SelVerbose makes the Selmer computation print more to the
// terminal about what it's doing. SelRamQuo makes the Selmer
// computation also supply the images of the inertia group in F_p*
// under the quotient rho*: G_K \to V/l. The order of the lists
// matches the order of the lines, allowing one to identify which
// nearly-ordinary Selmer groups correspond to unramified quotients.
// USeGRH allows one to specify whether to use class group bounds
// predicted by the general Riemann hypothesis, which often speeds
// up the computation.
SelRanks:=function(f,p,K : Ramification:=[], SelVerbose:=false,
  SelRamQuo:=false, UseGRH:=true)

  _<x>:=PolynomialRing(Integers());

  ZK:=MaximalOrder(K);

  L:=NumberField(f);

  assert IsSubfield(K,L);
  // L is the field such that Gal(L/K) = Im(rho)
  if #Ramification eq 0 then
    ZL:=MaximalOrder(L);
  else
    ZL:=MaximalOrder(L : Ramification:=Ramification cat
      PrimeFactors(Discriminant(K)));
  end if;
```

```
// the next few blocks compute the right modulus for our ray
// class computation
D:=Decomposition(ZL,p);

// this should be the structure of V as an abelain group
G:=AbelianGroup(ElementaryAbelianGroup(GrpGPC,p,2));

// we also need the primes in L over individual primes in K
// that lie over p. when K = Q this should be a list with one
// sub-list , etc
facK:=Factorization(p*ZK);
radL:=[[PP[1] : PP in ff] : ff in
  [Factorization(Parent(1*ZL)!u[1]) : u in facK]];

n:=#G;
v := Valuation(n,p);

ee:=Max([u[2] : u in Factorization(p*ZL)]);
// this is the exponent we will need
max_exp:=Ceiling(ee*v + ee/(p-1)+1);
I:=(&*[&*u : u in radL])^max_exp;

if UseGRH then
  SetClassGroupBounds("GRH");
end if;

// when max_exp is big this can take a long time.
time R, m :=RayClassGroup(I,[1..#RealPlaces(L)]);

// here is our maximal (Z/p)^n field , which contains all the
// elements of our Selmer group
mm:=MaxlPExt(R,m,p);
A:=AbelianExtension(mm);
```

```
time NSF,ACT,AUT,transfer:=NormalSubfields_K(A,Invariants(G),K);
print #NSF, "subfields to check";

// tells us what action each element of NSF encodes
group_stats,group_list:=ProfileNSF(ACT,p);
if SelVerbose then
group_stats;
end if;

// we need to choose a particular instance of the representation
// so that we can spot it in extensions
reps:=galreps(RelativeField(K,L),K,p);
ranks:=[];
rams:=[];
for rho in reps do
  tt,ff:=IsIsomorphic(AUT,Domain(rho[1]));
  // we precompose rho here for compatibility reasons
  // sometimes rho itself will not recognise AUT as its
  // domain, but a group isomorphic to it.
  rep:=<ff*rho[1],ff*rho[2],rho[3],rho[4]>;
  traces:=[Trace(rep[1](Domain(rep[1]).i)) : i in
    [1..Ngens(AUT)]];

  // packages all of the info into one easy-to-use source
  // we will be messing around with the condition over p, so
  // SelNoP is literally "Selmer with no condition on p"
  SelNoP:=rec<SEL | NSF:=NSF,
  ACT:=ACT,
  AUT:=AUT,
  transfer:=transfer,
  group_list:=group_list,
  p:=p,
    L:=L,
    K:=K,
    A:=A,
    mm:=mm,
    rad:=radL,
    traces:=traces,
    rep:=rep
    >;

    if SelRamQuo then
      RR,ram:=REL_NO_UR(SelNoP : Verbose:=SelVerbose,
         Fast:=true, RamQuo:=SelRamQuo);
      Append(~rams,ram);
    else
      RR:=REL_NO_UR(SelNoP : Verbose:=SelVerbose, Fast:=true,
        RamQuo:=SelRamQuo);
    end if;
    Append(~ranks,RR);
    frob_traces:=ComputeFrob(ZL,K,rho, 50);
    print "Ranks:";
    print RR;
    print "Traces of Frobenius:";
    print frob_traces;
  end for;

  return RR,rams;
end function;
```

A.2.7 H1H2_testing.m

```
TestSubgroupsOfGL2p:=function(p)
  G:=GL(2,p);
  S:=Subgroups(G);

  for i in [2..#S] do
    H:=S[i]`subgroup;
    M:=GModule(H);
    C:=CohomologyModule(H,M);
    dims:=[CohomologicalDimension(C,i) : i in [1,2]];
    if not dims eq [0,0] then
      i;
      GroupName(H);
      dims;
      "";
    end if;
  end for;
end function;
```

Appendix B

CODE FOR MODULAR FORMS

B.1 First cohomology

Remark B.1.1. Much of this code was adapted from code originally written by Haluk Şengün. To point out every place where code has been changed would be completely impractical; it suffices to say that his influence on the code was large and holistic.

B.1.1 `GetFormVals.m`

```
GetFormVals:=function(space,HECKEMATS,HECKEPRIMES,g)
  //gathers the eigenvectors
  der:=space`der;
  inn:=space`inn;

  mforms,pi:=quo<der|inn>;
  g:=Inverse(pi);

  EVs,list:=GET_EV(HECKEMATS);

  //gathers the generators of the hecke operators
  gens:=[];
  for i in HECKEPRIMES do
    t,gen:=IsPrincipal(i);
    Append(~gens,gen);
  end for;

  EV_systems:=[*<EVs[i,2],EVs[i,3]> : i in [1..#EVs]*];

  form_vals:=[];
  for i in [1..#EVs] do
    e:=EVs[i];
    if IsRationalSystem(e[3]) then
      if i gt 1 then
        offset:=&+[EVs[j,2] : j in [1..i-1]];
        else
        offset:=0;
      end if;
      for j in [1..e[2]] do
        Append(~form_vals,[*g(list[offset+j]),e[3],gens*]);
      end for;
    end if;
  end for;

  return EV_systems,form_vals;
end function;
```

B.1.2 `Hecke.m`

```
//TP is the prime ideal for the Hecke operator
HECKE:=function(TP,DIMdata)

  JJ:=DIMdata`level;
  coeff_size:=DIMdata`coeff_dim;
  char:=DIMdata`char;
```

```
weight:=DIMdata`weight cat DIMdata`det_twists;



DER:=DIMdata`der;
INN:=DIMdata`inn;

d:=DIMdata`d;
K:=QuadraticField(-d);
O<z>:=MaximalOrder(K);

A,Ai,B,U,Ui,J:=StandardMats(d);
t,s:=IsPrincipal(TP);
D:=Matrix(O,2,2,[s,0,0,1]);

if char eq 0*O then
  F:=K;
else
  F<t>:=ResidueClassField(char);
end if;

PD:=ProjMat(F,JJ,D,DIMdata`PL,DIMdata`r,DIMdata`chi);
TD:=RecursiveMatrix(PD,ModuleMat(char,weight,D));

// time saved here by computing coset reps and permutations once
cc:=CosetMats(K,TP);

CosetPerm:=function(reps,mat,I) //I is the level defining reps
  perm:=[];
  for R in reps do
    M:=R*mat;
  for i in [1..#reps] do
     conj:=M*reps[i]^(-1);
     if conj[2,1] mod I eq 0 and conj in Parent(M) then
```

```
        Append(~perm, i);
      end if;
      end for;
    end for;
    return perm;
end function;

permA:=CosetPerm(cc,A,TP);
permB:=CosetPerm(cc,B,TP);
permU:=CosetPerm(cc,U,TP);
permJ:=CosetPerm(cc,J,TP);

// this algebra is used for the polynomials
SSS:=MatrixAlgebra(F,coeff_size);

// X,Y,Z,W represent f(A), f(B), f(U), f(J) coefficients are matrices
PRR<X,Y,Z,W>:=PolynomialRing(SSS,4);

poly_A:=PRR!0;
poly_B:=PRR!0;
poly_U:=PRR!0;
poly_J:=PRR!0;

// we will get the summation that defines the Hecke operator's
// image on A, B, U and J
for i in [1..Norm(TP)+1] do
  poly_A:= PRR!poly_A + PRR!POLY(A,permA,cc,TP,i,DIMdata,TD,d);
  poly_B:= PRR!poly_B + PRR!POLY(B,permB,cc,TP,i,DIMdata,TD,d);
  poly_U:= PRR!poly_U + PRR!POLY(U,permU,cc,TP,i,DIMdata,TD,d);
  poly_J:= PRR!poly_J + PRR!POLY(J,permJ,cc,TP,i,DIMdata,TD,d);
end for;

H11:=MonomialCoefficient(poly_A,X);
```

```
H21:=MonomialCoefficient(poly_A,Y);
H31:=MonomialCoefficient(poly_A,Z);
H41:=MonomialCoefficient(poly_A,W);

C1:=VerticalJoin([H11,H21,H31,H41]);

H12:=MonomialCoefficient(poly_B,X);
H22:=MonomialCoefficient(poly_B,Y);
H32:=MonomialCoefficient(poly_B,Z);
H42:=MonomialCoefficient(poly_B,W);

C2:=VerticalJoin([H12,H22,H32,H42]);

H13:=MonomialCoefficient(poly_U,X);
H23:=MonomialCoefficient(poly_U,Y);
H33:=MonomialCoefficient(poly_U,Z);
H43:=MonomialCoefficient(poly_U,W);

C3:=VerticalJoin([H13,H23,H33,H43]);

H14:=MonomialCoefficient(poly_J,X);
H24:=MonomialCoefficient(poly_J,Y);
H34:=MonomialCoefficient(poly_J,Z);
H44:=MonomialCoefficient(poly_J,W);

C4:=VerticalJoin([H14,H24,H34,H44]);


H0:=HorizontalJoin([C1,C2,C3,C4]);

R, f:=quo<DER|INN>;

B:=Basis(R);
```

```
  pB:=[];
  g:=Inverse(f);

  for b in B do
    Append(~pB,g(b));
  end for;

  Mat:=[];

  for b in pB do
    Append(~Mat,ElementToSequence(f(b*H0)));
  end for;

  return Matrix(F,Mat);
end function;
```

B.1.3 `Hecke_poly.m`

```
POLY:=function(M,permM,reps,TP,i,DIMdata,TD,d)

  JJ:=DIMdata`level;
  TA:=DIMdata`ta;
  TB:=DIMdata`tb;
  TU:=DIMdata`tu;
  TJ:=DIMdata`tj;
  coeff_size:=DIMdata`coeff_dim;
  char:=DIMdata`char;
  weight:=DIMdata`weight cat DIMdata`det_twists;

  K:=QuadraticField(-d);
  O<z>:=MaximalOrder(K);
  Z:=Integers();
```

```
A,Ai,B,U,Ui,J:=StandardMats(d);

if char eq 0*O then
  F:=K;
else
  F<t>:=ResidueClassField(char);
end if;

t,s:=IsPrincipal(TP);
D:=Matrix(K,2,2,[s,0,0,1]);

SSS:=MatrixAlgebra(F,coeff_size);
N:=SSS!0;
ID:=SSS!1;

PRR<X,Y,Z,W>:=PolynomialRing(SSS,4);
IDD:=Matrix(O,2,2,[1,0,0,1]);

hM:=D*reps[i]*M*reps[permM[i]]^(-1)*D^(-1);

g:=PRR!0;

if hM eq IDD then
   return(g);
end if;
if hM eq -IDD then
   return(g);
end if;

decomp:=WORD(hM,d);

t_decomp:=[];
```

```
//creates the same decomposition, but with the T matrices
for i in [1..#decomp] do
  if decomp[i,1] eq A then
    Append(~t_decomp,MatPow(TA,decomp[i,2]));
  elif decomp[i,1] eq B then
  A ppend(~t_decomp,MatPow(TB,decomp[i,2]));
  elif decomp[i,1] eq U then
    Append(~t_decomp,MatPow(TU,decomp[i,2]));
  elif decomp[i,1] eq J then
    Append(~t_decomp,MatPow(TJ,decomp[i,2]));
  else
    Append(~t_decomp,"you stop that");
  end if;
end for;

//does the fiddly cocycle calculating
for i in [1..#decomp] do
  if decomp[i][1] eq A then
    g+:=HeckeMatP(N,TA,decomp[i][2])*&*t_decomp[i+1..#t_decomp]*X;
  elif decomp[i][1] eq B then
    g+:=HeckeMatP(N,TB,decomp[i][2])*&*t_decomp[i+1..#t_decomp]*Y;
  elif decomp[i][1] eq U then
    g+:=HeckeMatP(N,TU,decomp[i][2])*&*t_decomp[i+1..#t_decomp]*Z;
  elif decomp[i][1] eq J then
    g+:=HeckeMatP(N,TJ,decomp[i][2])*&*t_decomp[i+1..#t_decomp]*W;
  else
    g:="you stop that";
  end if;
end for;

Rj:=reps[permM[i]];
PRj:=ProjMat(F,JJ,Rj,DIMdata`PL,DIMdata`r,DIMdata`chi);
```

```
  TRj:=RecursiveMatrix(PRj,ModuleMat(char,weight,Rj));

  return MonomialCoefficient(g,X)*TD*TRj*X+
  MonomialCoefficient(g,Y)*TD*TRj*Y+
  MonomialCoefficient(g,Z)*TD*TRj*Z+
  MonomialCoefficient(g,W)*TD*TRj*W;
end function;
```

B.1.4 `LevelData_pgl.m`

```
ToSpace:= recformat <
  level               : RngOrdIdl,
  projmatA            : List,
  projmatAi           : List,
  projmatB            : List,
  projmatU            : List,
  projmatUi           : List,
  projmatJ            : List,
  d                   : RngIntElt,
  char                : RngQuadIdl,
  PL                  : SetIndx,
  r                   : UserProgram,
  chi                 : GrpDrchNFElt >;


LEVELDATA:=function(level,d,char,chi)

  K:=QuadraticField(-d);
  O<z>:=MaximalOrder(K);

  char:=char*O;

  if char eq 0*O then
```

```
    F:=K;
  else
    char:=Factorization(char)[1,1];
    F<t>:=ResidueClassField(char);
  end if;

  PL,r:=ProjectiveLine(quo<O|level>);

  A,Ai,B,U,Ui,J:=StandardMats(d);

  DG:=DirichletGroup(level);
  chi:=Elements(DG)[chi];

  if level ne 1*O then
    PA:=ProjMat(F,level,A,PL,r,chi);
    PAi:=ProjMat(F,level,Ai,PL,r,chi);
    PB:=ProjMat(F,level,B,PL,r,chi);
    PU:=ProjMat(F,level,U,PL,r,chi);
    PUi:=ProjMat(F,level,Ui,PL,r,chi);
    PJ:=ProjMat(F,level,J,PL,r,chi);
  else
    PA:=[**];
    PAi:=[**];
    PB:=[**];
    PU:=[**];
    PUi:=[**];
    PJ:=[**];
  end if;

  Data:=rec< ToSpace | level:=level, projmatA:=PA, projmatAi:=PAi,
      projmatB:=PB, projmatU:=PU, projmatUi:=PUi, projmatJ:=PJ,
      d:=d, char:=char, PL:=PL, r:=r, chi:=chi  >;
  return Data;
end function;
```

B.1.5 `loading_script_pgl.m`

```
AttachSpec("ArtinAlgebras/ArtinAlgebras.spec");

load "ProjAction.m";
load "ModuleAction.m";
load "RecursiveMatrix.m";
load "Utility.m";


load "LevelData_pgl.m";
load "space_mat.m";
load "Space_pgl.m";

load "Hecke_poly.m";
load "Hecke.m";
load "Utility_Hecke.m";
load "GetFormVals.m";

load "Periods.m";


//
// Change things here
//

//field of definition
d:=1;
K:=QuadraticField(-d);
```

```
O<z>:=MaximalOrder(K);

//level, weight, twists (TBA), number of eigenvalues, dirichlet character
N:=-4*z + 3;
level:=N*O;
wt:=[0,0];
tw:=[0,0];
chi:=1;
HB:=40;
char:=0;

LD:=LEVELDATA(level,d,char,chi);
space:=DIM(LD,wt[1],wt[2],tw[1],tw[2]);

print "Full dimension:", space`dim;
print "Cocycle dimension:", Dimension(space`der);
print "Coboundary dimension:", Dimension(space`inn);

print "Eisenstein dimension:", CuspNumber_SL(K,level);

der:=space`der;
inn:=space`inn;

mforms,pi:=quo<der|inn>;
g:=Inverse(pi);

if char eq 0 then
  HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level) eq 1*O];
  //HP:=[TP : TP in PrimesUpTo(HB,K)];
else
  HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level*char) eq 1*O];
end if;
HNF:=[HNF_basis(J): J in HP];
```

```
ParallelSort(~HNF,~HP);

time HH:= GetHeckeMatrices(space,HP);

EV_systems,form_vals:=GetFormVals(space,HH,HP,g);

EV_systems;
```

B.1.6 `ModuleAction.m`

```
ModuleMat:=function(char,weight,M)

  R:=CoefficientRing(Parent(M));
  K:=NumberField(R);

  if char eq 0*R then
  F:=K;
  down:=IdentityHomomorphism(R); //compatibility
  else
  F<t>,down:=ResidueClassField(char);
  end if;

  P<x,y>:=PolynomialRing(F,2);
  p:=Characteristic(F);

  k:=weight[1];
  l:=weight[2];
  d1:=weight[3];
  d2:=weight[4];

  Symm:=function(k,d,T)
    ST:=ZeroMatrix(F,k+1,k+1);
    for i in [0..k] do
```

```
      Q:=(down(T[1,1])*x+down(T[1,2])*y)^(k-i)*(down(T[2,1])*x+
        down(T[2,2])*y)^(i);
      for j in [0..k] do
        ST[i+1,j+1]:=MonomialCoefficient(Q,x^(k-j)*y^(j));
      end for;
    end for;
    return down(Determinant(T))^d*ST;
  end function;

  Mc:=Matrix(R,2,2,[R!Conjugate(M[1][1]),R!Conjugate(M[1][2]),
    R!Conjugate(M[2][1]),R!Conjugate(M[2][2])]);

  TM:=TensorProduct(Symm(k,d1,M),Symm(l,d2,Mc));

  return TM;
end function;
```

B.1.7 Periods.m

```
// Returns 1+M+M^2+...+M^(m-1) (or the correct version for an inverse)
MatP:=function(M,m)
  Z:=Integers();
  W:=M-M;
  if m lt 0 then
    for j in [0..(-m-1)] do
      W:=W+M^j;
    end for;
    W:= -W*M^Z!(m);
  end if;
  if m gt 0 then
    for j in [0..Z!(m-1)] do
      W:=W+M^j;
    end for;
  end if;
  return W;
end function;


// Returns a matrix in Gamma0(N) which takes 0 to k/p
PathMat:=function(K,N,k,p)

  O<z>:=MaximalOrder(K);
  J:=N*O;
  ord:=#UnitGroup(quo<O|J>);

  a:=p^(ord-1) mod N;

  t:=(a*p-1)/N;

  coeffs:=Bezout(p,k);

  u:=-t*coeffs[1]/GCD(p,k);
  v:=t*coeffs[2]/GCD(p,k);

  M:=Matrix(O,2,2,[a+u*N,k,v*N,p]);
  assert Determinant(M) eq 1;
  assert M[2,1] mod N eq 0;

  return M;
end function;


// Given f a cocycle and W a matrix, returns f(w)
EvaluateCocycle:=function(space,f,W)

  K:=space`field;
```

```
O<z>:=MaximalOrder(K);

d:=space`d;
A,Ai,B,U,Ui,J:=StandardMats(d);

der:=space`der;
inn:=space`inn;

TA:=space`ta;
TB:=space`tb;
TU:=space`tu;
TJ:=space`tj;

n:=space`coeff_dim;

f_A:=[];
f_B:=[];
f_U:=[];
f_J:=[];

// fills the f_A, f_B, f_U vectors with their respective parts
// of the full vector of f
for i in [1..n] do
  Append(~f_A,f[i]);
  Append(~f_B,f[n+i]);
  Append(~f_U,f[2*n+i]);
  Append(~f_J,f[3*n+i]);
end for;

f_A:=Vector(n,f_A);
f_B:=Vector(n,f_B);
f_U:=Vector(n,f_U);
f_J:=Vector(n,f_J);
```

```
Z:=Integers();

decomp:=WORD(W,d);
// creating the decomposition e.g. [A,3],[B,1],[U,-4],[B,1]
// -> TA^3,TB,TU^-4,TB
t_decomp:=[];
for i in [1..#decomp] do
  if decomp[i][1] eq A then
    Append(~t_decomp,TA^(Z!decomp[i][2]));
  elif decomp[i][1] eq B then
    Append(~t_decomp,TB^(Z!decomp[i][2]));
  elif decomp[i][1] eq U then
    Append(~t_decomp,TU^(Z!decomp[i][2]));
  elif decomp[i][1] eq J then
    Append(~t_decomp,TJ^(Z!decomp[i][2]));
  else
    print "you stop that", "t_decomp";
  end if;
end for;

image:=f_A-f_A; //zero vector
N:=Parent(TA)!0; // zero matrix

//adding on each term in the cocycle decomposition
for i in [1..#decomp] do
  if decomp[i][1] eq A then
    image+:=f_A*MatP(TA,decomp[i][2])*&*t_decomp[i+1..#t_decomp];
  elif decomp[i][1] eq B then
    image+:=f_B*MatP(TB,decomp[i][2])*&*t_decomp[i+1..#t_decomp];
  elif decomp[i][1] eq U then
    image+:=f_U*MatP(TU,decomp[i][2])*&*t_decomp[i+1..#t_decomp];
  elif decomp[i][1] eq J then
```

```
      image+:=f_J*MatP(TJ,decomp[i][2])*&*t_decomp[i+1..#t_decomp];
    else
      print "you stop that","decomp";
    end if;
  end for;

  return image;
end function;


// Calculates the period sum f(M_a) for a in Z_K/P
PeriodAverage:=function(space,form,r)

  d:=space`d;
  K:=QuadraticField(-d);
  O<z>:=MaximalOrder(K);
  level:=space`level;
  t,N:=IsPrincipal(level);
  p:=form[3][r];
  // integral from 0 to 0 doesn't contribute, so we ignore it
  ModPoints:=Exclude(Elements(quo<O|p*O>),0);

  mats:=[];
  images:=[];
  image:=EvaluateCocycle(space,form[1],Matrix(O,2,2,[1,0,0,1]));

  for k in ModPoints do
    Append(~mats,PathMat(K,N,k,p));
    Append(~images,EvaluateCocycle(space,form[1],mats[#mats]));
    image+:=images[#images];
  end for;
  R:=Parent(form[1][1]);
  if R!(1+Norm(O!p)-form[2][r]) ne 0 then
```

```
    return form[3][r],"good", image[space`id_index]/(1+Norm(O!p)-form[2][r]),
      [images[i][space`id_index] : i in [1..#images]];
  else
    return form[3][r], "bad", image[space`id_index],
      [images[i][space`id_index] : i in [1..#images]];
  end if;
end function;
```

B.1.8 ProjAction.m

```
// For integers x & y with GCD(x,y) = d, gives integers [a,b]
// such that ax+by = d
Bezout:=function(x,y)
  Q:=Rationals();
  if y eq 0 then
    return [Sign(Q!x),0];
  end if;
  q1:=x div y;
  r:= x - q1*y;
  if r eq 0 then
    if Norm(q1) ge 0 then
      return [1 , 1 - q1];
    else
      return [1 , -1 - q1];
    end if;
  else
    q2:=$$(y,r);
    return [q2[2],q2[1]-q2[2]*q1];
  end if;
end function;


ProjMat:=function(K,J,M,PL,r,chi)
```

```
O:=CoefficientRing(Parent(M));

// sometimes M is defined over K rather than O, this deals with
// that case
if Type(O) ne RngQuad then
  O:=MaximalOrder(O);
end if;

SS:=MatrixAlgebra(O,2);
M:=SS!M;
FacMatrix:=[**];

proj_action:=function(M,i)
  t,im,sca :=r(PL[i]*M,true,true);
  if t then
    return [*Index(PL,im),sca*];
  else
    return [*-1,0*];
  end if;
end function;

perm:=[];
scalars:=[];
for i in [1..#PL] do
  PA:=proj_action(M,i);
  Append(~perm,PA[1]);
  Append(~scalars,PA[2]);
end for;

l:=#PL;
Mat:=[];
for i in [1..l] do
```

```
    new_row:=[];
    for j in [1..l] do
      if j eq perm[i] then
        Append(~new_row,chi(scalars[i]));
      else
        Append(~new_row,0);
      end if;
    end for;
    Append(~Mat,new_row);
  end for;

  Append(~FacMatrix,ChangeRing(Matrix(Mat),K));

  Insert(~FacMatrix,1,#PL);

  return FacMatrix;
end function;


// Computes coset representative matrices for Gamma0(J) in PSL(Z_K)
CosetMats:=function(K,J)

  O<z>:=MaximalOrder(K);
  PL,r:=ProjectiveLine(quo<O|J>);
  t,j:=IsPrincipal(J);

  coset_reps:=[];

  for i in [1..#PL] do
    bottomRow:=[PL[i][1],PL[i][2]];
    // moving to a rep with gcd = 1
    while Abs(Norm(GCD(bottomRow[1],bottomRow[2]))) ne 1 do
      bottomRow[1]+:=j;
    end while;
    topRow:=Bezout(bottomRow[1],bottomRow[2]);
    newMatrix:=Matrix(O,2,2,[topRow[2],-topRow[1],bottomRow[1],
      bottomRow[2]]);
    u:=Determinant(newMatrix);
    if u ne 1 then /*this makes the determinant 1 always */
      newMatrix:=Matrix(O,2,2,[newMatrix[1][1]/u,
        newMatrix[1][2]/u,newMatrix[2][1],newMatrix[2][2]]);
    end if;
    assert Determinant(newMatrix) eq 1;
    Append(~coset_reps,newMatrix);
  end for;

return coset_reps;
end function;



// Computes the matrices t_i(M) and coset representatives for Gamma_0(P),
// where P is a prime ideal giving the Hecke operator
CoresMat:=function(K,P,M)
  O<z>:=MaximalOrder(K);
  SS:=MatrixAlgebra(O,2);
  PL,r:=ProjectiveLine(quo<O|P>);
  level:=#PL;
  coset_reps:=[**];
  permuted_reps:=[**];

  proj_action:=function(M,i)
    t,im :=r(PL[i]*(SS!M),true,false);
    return Index(PL,im);
  end function;

  coset_reps:=CosetMats(K,P);
  for i in [1..#PL] do
    t_iM:=coset_reps[proj_action(M,i)];
    Append(~permuted_reps,t_iM);
  end for;

  return [*coset_reps, permuted_reps *];
end function;


CoresMatAlt:=function(K,P,M,space)
  O<z>:=MaximalOrder(K);
  SS:=MatrixAlgebra(O,2);
  PL:=space`PLlevel;
  r:=space`rpl;
  level:=#PL;
  coset_reps:=[**];
  permuted_reps:=[**];

  proj_action:=function(M,i)
    t,im :=r(PL[i]*(SS!M),true,false);
    return Index(PL,im);
  end function;

  coset_reps:=CosetMats(K,P);

  for i in [1..#PL] do
    t_iM:=coset_reps[proj_action(M,i)];
    Append(~permuted_reps,t_iM);
  end for;

  return [*coset_reps, permuted_reps *];
end function;
```

```
// Returns the index of (0,1) in Proj Line
PLIndex:=function(K,level)
  O<z>:=MaximalOrder(K);
  PL,r:=ProjectiveLine(quo<O|level*O>);

  pos:=-1;
  for i in [1..#PL] do
    if PL[i][1] eq 0 and PL[i][2] eq 1 then
      pos:=i;
      break i;
    end if;
  end for;
  return pos;
end function;
```

B.1.9 `RecusriveMatrix.m`

```
RecursiveMatrix:=function(MatrixList,A)
  // the first entry of the MatrixList is supposed to be the size
  // of the full projective line
  M:=TensorProduct(MatrixList[2],A);
  for j in [3..#MatrixList] do
    M:=TensorProduct(MatrixList[j],M);
  end for;
  return Matrix(M);
end function;
```

B.1.10 `space_mat.m`

```
space_mat_pgl:=function(d,TA,TAi,TB,TU,TUi,TJ)
  ID:=TA^0;
```

```
N:=ID-ID;

TAB2:=TA*TB*TA*TB;
TAB:=TA*TB;

if d eq 1 then

   TU2:=TU^2;
   TJ2:=TJ^2;
   TJi:=TJ^(-1);

   E1:=VerticalJoin([N,TB+ID,N,N]);
   E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N,N]);
   E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi,N]);
   E4:=VerticalJoin([N, TU * TB * TUi * TB * TU * TB * TUi * TB *
      TU * TB * TUi + TUi * TB  * TU * TB * TUi * TB * TU * TB
      * TUi + TU * TB * TUi * TB * TU * TB * TUi + TUi * TB * TU
      * TB * TUi + TU * TB * TUi + TUi, TB * TUi * TB * TU * TB *
      TUi * TB * TU * TB * TUi - TUi * TB * TU * TB * TUi * TB *
      TU * TB * TUi + TB * TUi * TB * TU * TB * TUi - TUi * TB *
      TU * TB * TUi + TB * TUi - TUi, N ]);
   E5:=VerticalJoin([N, TU2 * TB * TUi * TB * TU2 * TB * TUi + TUi *
      TB * TU2 * TB * TUi + TU2 * TB * TUi + TUi, TU * TB * TUi *
      TB * TU2 * TB * TUi + TB * TUi * TB * TU2 * TB * TUi - TUi *
      TB * TU2 * TB * TUi + TU * TB * TUi + TB * TUi - TUi, N]);
   E6:=VerticalJoin([ TU * TB * TA * TUi * TB * TA * TU * TB * TA *
      TUi * TB + TUi * TB * TA *  TU * TB * TA * TUi * TB + TU *
      TB * TA * TUi * TB + TUi * TB, TA * TUi * TB * TA * TU * TB *
      TA * TUi * TB + TA * TU * TB * TA * TUi * TB + TA * TUi *
      TB + ID, TB * TA * TUi * TB * TA * TU * TB * TA * TUi * TB -
      TUi * TB * TA * TU * TB * TA * TUi * TB + TB * TA * TUi *
      TB - TUi * TB, N]);
```

```
  J0:=VerticalJoin([N, N, N, TJ*TJ2 + TJ2 + TJ + ID]);
  J1:=VerticalJoin([ -TAi * TJi * TU, N, ID, TAi * TJi * TU - TJi
    * TU  ]);
  J2:=VerticalJoin([ ID, N, TJi * TA, TU * TJi * TA - TJi * TA ]);
  J3:=VerticalJoin([ N, TJ * TB * TJ * TB * TJ * TB * TJ + TJ *
    TB * TJ * TB * TJ + TJ * TB * TJ + TJ, N, (TB * TJ)^3 +
    (TB * TJ)^2 + TB * TJ + ID ]);

  E:=HorizontalJoin([E1,E2,E3,E4,E5,E6,J0,J1,J2,J3]);

elif d eq 2 then

  E1:=VerticalJoin([N,TB+ID,N,N]);
  E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N,N]);
  E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi,N]);
  E4:=VerticalJoin([ N, TUi * TB * TU * TB * TUi * TB * TU + TU *
  TB * TUi * TB * TU + TUi* TB * TU + TU, -TUi * TB * TU * TB *
  TUi * TB * TU + TB * TUi * TB * TU - TUi* TB * TU + ID, N ]);

  J0:=VerticalJoin([N,N,N,ID+TJ]);
  J1:=VerticalJoin([ TJ * TA + ID, N, N, TA * TJ * TA + TA ]);
  J2:=VerticalJoin([ N, TJ * TB + ID, N, TB * TJ * TB + TB ]);
  J3:=VerticalJoin([ N, N, TJ * TU + ID, TU * TJ * TU + TU ]);

  E:=HorizontalJoin([E1,E2,E3,E4,J0,J1,J2,J3]);

elif d eq 3 then

  TJi:=TJ^(-1);

  E1:=VerticalJoin([N,TB+ID,N,N]);
  E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N,N]);
  E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi,N]);
```

```
E4:=VerticalJoin([  TUi * TA * TB * TU * TB * TUi * TA * TUi *
   TA * TB + TB * TU * TB * TUi * TA * TUi * TA * TB + TUi * TA *
   TB + TB, TUi * TA * TUi * TA * TB * TU * TB * TUi * TA * TUi *
   TA * TB + TU * TB * TUi * TA * TUi * TA * TB + TUi * TA * TUi
   * TA * TB + ID, TB * TUi * TA * TUi * TA * TB * TU * TB * TUi
   * TA * TUi * TA * TB - TUi * TA * TUi * TA * TB * TU * TB *
    TUi * TA * TUi * TA * TB - TUi * TA * TB * TU * TB * TUi *
    TA *TUi * TA * TB + TB * TUi * TA * TUi * TA * TB - TUi *
    TA * TUi * TA * TB - TUi * TA * TB, N  ]);
E5:=VerticalJoin([TB * TU * TB * TUi * TA * TB * TU * TB * TUi
   * TA * TB + TB * TU * TB * TUi * TA * TB + TB, TUi * TA *
   TB * TU * TB * TUi * TA * TB * TU * TB * TUi * TA * TB +
   (TU * TB * TUi * TA * TB)^2 + TUi * TA * TB * TU * TB *
   TUi * TA * TB + TU * TB * TUi * TA * TB + TUi * TA * TB +
   ID, TB * TUi * TA * TB * TU * TB * TUi * TA * TB * TU *
   TB * TUi * TA * TB - TUi * TA * TB * TU * TB * TUi * TA
   * TB * TU * TB * TUi * TA * TB + TB * TUi * TA * TB * TU *
   TB * TUi * TA * TB - TUi * TA * TB * TU * TB * TUi * TA *
   TB + TB * TUi * TA * TB - TUi * TA * TB, N ]);
E6:=VerticalJoin([ TU * TB * TUi * TA * TB * TAi * TU * TB *
   TAi * TU * TB * TUi * TA * TB + TB * TAi * TU * TB * TAi * TU
   * TB * TUi * TA * TB - TAi * TU * TB * TAi * TU * TB * TUi *
   TA * TB - TAi * TU * TB * TUi * TA * TB + TB, TUi * TA * TB
   * TAi * TU * TB * TAi * TU * TB * TUi * TA * TB + TAi *
   TU * TB * TAi * TU * TB * TUi * TA * TB + TAi * TU * TB *
   TUi * TA * TB + TUi * TA * TB + ID, TB * TUi * TA * TB *
   TAi * TU * TB * TAi * TU * TB * TUi * TA * TB - TUi * TA *
   TB * TAi * TU * TB * TAi * TU * TB  * TUi * TA * TB + TB *
   TAi * TU * TB * TUi * TA * TB + TB * TUi * TA * TB -
   TUi * TA * TB, N ]);

J0:=VerticalJoin([N, N, N, TJ^5 + TJ^4 + TJ^3 + TJ^2 + TJ + ID]);
J1:=VerticalJoin([  TJi * TUi, N, -TUi, TA * TJi * TUi - TJi
    * TUi  ]);
  J2:=VerticalJoin([ ID, N, TJi * TUi * TA - TUi * TA, TU * TJi
    * TUi * TA - TJi * TUi * TA  ]);
  J3:=VerticalJoin([ TB * TU * TB * TUi * TB * TAi * TB - TAi *
    TB, TJi * TB * TA * TB * TU * TB * TUi * TB * TAi * TB +
    TA * TB * TU * TB * TUi * TB * TAi * TB + TU * TB * TUi
    * TB * TAi * TB + TUi * TB * TAi * TB + TAi * TB + ID,
    TB * TUi * TB * TAi * TB - TUi * TB * TAi * TB, TB * TJi
    * TB * TA * TB * TU * TB * TUi * TB * TAi * TB - TJi *
    TB * TA * TB * TU * TB * TUi * TB * TAi * TB  ]);

  E:=HorizontalJoin([E1,E2,E3,E4,E5,E6,J0,J1,J2,J3]);

elif d eq 7 then

  E1:=VerticalJoin([N,TB+ID,N,N]);
  E2:=VerticalJoin([TB*(TAB2+TAB+ID),(TAB2+TAB+ID),N,N]);
  E3:=VerticalJoin([(TAi*(ID-TUi)),N,(TAi-ID)*TUi,N]);
  E4:=VerticalJoin([ TUi * TB * TU * TB * TA * TUi * TB *
  TU + TUi*TB*TU, TA * TUi * TB * TU * TB * TA * TUi * TB *
  TU + TU * TB * TA * TUi * TB * TU + TA * TUi * TB * TU +
  TU, -TUi * TB * TU * TB * TA * TUi * TB * TU + TB * TA *
  TUi * TB * TU - TUi*TB*TU + ID,N ]);

  J0:=VerticalJoin([N,N,N,ID+TJ]);
  J1:=VerticalJoin([ TJ * TA + ID, N, N, TA * TJ * TA + TA ]);
  J2:=VerticalJoin([ N, TJ * TB + ID, N, TB * TJ * TB + TB ]);
  J3:=VerticalJoin([ N, N, TJ * TU + ID, TU * TJ * TU + TU ]);

  E:=HorizontalJoin([E1,E2,E3,E4,J0,J1,J2,J3]);

elif d eq 11 then
```

```
    E1:=VerticalJoin([N,TB+ID,N,N]);
    E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N,N]);
    E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi,N]);
    E4:=VerticalJoin([ TUi * TB * TU * TB * TA * TUi * TB * TU *
      TB * TA * TUi * TB * TU +TUi * TB * TU * TB * TA * TUi *
      TB * TU + TUi*TB*TU, TA * TUi * TB * TU * TB * TA * TUi
      * TB * TU * TB * TA * TUi * TB * TU + TU * TB * TA * TUi
      * TB * TU * TB * TA * TUi * TB * TU + TA * TUi * TB *
      TU * TB * TA * TUi * TB *TU + TU * TB * TA * TUi * TB *
      TU + TA * TUi * TB * TU + TU, -TUi * TB * TU * TB * TA
      * TUi * TB * TU * TB * TA * TUi * TB * TU +
      (TB * TA * TUi * TB * TU)^2 - TUi * TB * TU * TB * TA *
      TUi * TB * TU + TB * TA * TUi * TB * TU - TUi*TB*TU + ID,
      N ]);

    J0:=VerticalJoin([N,N,N,ID+TJ]);
    J1:=VerticalJoin([ TJ * TA + ID, N, N, TA * TJ * TA + TA ]);
    J2:=VerticalJoin([ N, TJ * TB + ID, N, TB * TJ * TB + TB ]);
    J3:=VerticalJoin([ N, N, TJ * TU + ID, TU * TJ * TU + TU ]);

    E:=HorizontalJoin([E1,E2,E3,E4,J0,J1,J2,J3]);

  else
    return "you stop that", "space_mat_pgl";
  end if;

  return E;

end function;


space_mat_psl:=function(d,TA,TAi,TB,TU,TUi)
  ID:=TA^0;
```

```
N:=ID-ID;

TAB2:=TA*TB*TA*TB;
TAB:=TA*TB;

if d eq 1 then

  TU2:=TU^2;

  E1:=VerticalJoin([N,TB+ID,N]);
  E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N]);
  E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi]);
  E4:=VerticalJoin([N, TU * TB * TUi * TB * TU * TB * TUi
    * TB * TU * TB * TUi + TUi * TB * TU * TB * TUi * TB
     * TU * TB * TUi + TU * TB * TUi * TB * TU * TB * TUi
     + TUi * TB * TU * TB * TUi + TU * TB * TUi + TUi,
     TB * TUi * TB * TU * TB * TUi * TB * TU * TB * TUi
     - TUi * TB * TU * TB * TUi * TB * TU * TB * TUi + TB
      * TUi * TB * TU * TB * TUi - TUi * TB * TU * TB *
        TUi + TB * TUi - TUi ]);
  E5:=VerticalJoin([N, TU2 * TB * TUi * TB * TU2 * TB * TUi +
    TUi * TB * TU2 * TB * TUi + TU2 * TB * TUi + TUi,
    TU * TB * TUi * TB * TU2 * TB * TUi + TB * TUi * TB *
    TU2 * TB * TUi - TUi * TB * TU2 * TB * TUi + TU * TB *
    TUi + TB * TUi - TUi ]);
  E6:=VerticalJoin([ TU * TB * TA * TUi * TB * TA * TU * TB
    * TA * TUi * TB + TUi * TB * TA * TU * TB * TA * TUi
    * TB + TU * TB * TA * TUi * TB + TUi * TB, TA * TUi
    * TB * TA * TU * TB * TA * TUi * TB + TA * TU * TB
    * TA * TUi * TB + TA * TUi * TB + ID, TB * TA * TUi *
    TB * TA * TU * TB * TA * TUi * TB - TUi * TB * TA * TU
    * TB * TA * TUi * TB + TB * TA * TUi * TB - TUi * TB ]);
  E:=HorizontalJoin([E1,E2,E3,E4,E5,E6]);

elif d eq 2 then

  E1:=VerticalJoin([N,TB+ID,N]);
  E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N]);
  E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi]);
  E4:=VerticalJoin([ N, TUi * TB * TU * TB * TUi * TB * TU
    + TU * TB * TUi * TB * TU + TUi* TB * TU + TU, -TUi *
    TB * TU * TB * TUi * TB * TU + TB * TUi * TB * TU -
    TUi* TB * TU + ID ]);

  E:=HorizontalJoin([E1,E2,E3,E4]);

elif d eq 3 then

  E1:=VerticalJoin([N,TB+ID,N]);
  E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N]);
  E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi]);
  E4:=VerticalJoin([  TUi * TA * TB * TU * TB * TUi * TA * TUi
    * TA * TB + TB * TU * TB * TUi * TA * TUi * TA * TB + TUi
    * TA * TB + TB, TUi * TA * TUi * TA * TB * TU * TB *
    TUi * TA * TUi * TA * TB + TU * TB * TUi * TA * TUi *
    TA * TB + TUi * TA * TUi * TA * TB + ID, TB * TUi * TA
    * TUi * TA * TB * TU * TB * TUi * TA * TUi * TA * TB
    - TUi * TA * TUi * TA * TB * TU * TB * TUi * TA * TUi
    * TA * TB - TUi * TA * TB * TU * TB * TUi * TA * TUi *
    TA * TB + TB * TUi * TA * TUi * TA * TB - TUi * TA *
    TUi * TA * TB - TUi * TA * TB ]);
  E5:=VerticalJoin([TB * TU * TB * TUi * TA * TB * TU * TB *
    TUi * TA * TB + TB * TU * TB * TUi * TA * TB + TB,
    TUi * TA * TB * TU * TB * TUi * TA * TB * TU * TB *
    TUi * TA * TB + (TU * TB * TUi * TA * TB)^2 + TUi *
```

```
      TA * TB * TU * TB * TUi * TA * TB + TU * TB * TUi * TA
      * TB + TUi * TA * TB + ID, TB * TUi * TA * TB * TU * TB
      * TUi * TA * TB * TU * TB * TUi * TA * TB - TUi * TA *
       TB * TU * TB * TUi * TA * TB * TU * TB * TUi * TA *
       TB + TB * TUi * TA * TB * TU * TB * TUi * TA * TB
       - TUi * TA * TB * TU * TB * TUi * TA * TB + TB * TUi
       * TA * TB - TUi * TA * TB ]);
    E6:=VerticalJoin([ TU * TB * TUi * TA * TB * TAi * TU * TB *
      TAi * TU * TB * TUi * TA * TB + TB * TAi * TU * TB *
      TAi * TU * TB * TUi * TA * TB - TAi * TU * TB * TAi *
      TU * TB * TUi * TA * TB - TAi * TU * TB * TUi * TA *
      TB + TB, TUi * TA * TB * TAi * TU * TB * TAi * TU * TB
      * TUi * TA * TB + TAi * TU * TB * TAi * TU * TB * TUi *
      TA * TB + TAi * TU * TB * TUi * TA * TB + TUi * TA * TB
      + ID, TB * TUi * TA * TB * TAi * TU * TB * TAi * TU *
      TB * TUi * TA * TB - TUi * TA * TB * TAi * TU * TB *
      TAi * TU * TB * TUi * TA * TB + TB * TAi * TU * TB *
      TUi * TA * TB + TB * TUi * TA * TB - TUi * TA * TB ]);

    E:=HorizontalJoin([E1,E2,E3,E4,E5,E6]);

  elif d eq 7 then

    E1:=VerticalJoin([N,TB+ID,N]);
    E2:=VerticalJoin([TB*(TAB2+TAB+ID),(TAB2+TAB+ID),N]);
    E3:=VerticalJoin([(TAi*(ID-TUi)),N,(TAi-ID)*TUi]);
    E4:=VerticalJoin([ TUi * TB * TU * TB * TA * TUi * TB *
      TU + TUi*TB*TU, TA * TUi * TB * TU * TB * TA * TUi *
      TB * TU + TU * TB * TA * TUi * TB * TU + TA * TUi *
      TB * TU + TU, -TUi * TB * TU * TB * TA * TUi * TB *
      TU + TB * TA * TUi * TB * TU - TUi*TB*TU + ID ]);

    E:=HorizontalJoin([E1,E2,E3,E4]);
  elif d eq 11 then

    E1:=VerticalJoin([N,TB+ID,N,N]);
    E2:=VerticalJoin([TB*(TAB2+TAB+ID),TAB2+TAB+ID,N]);
    E3:=VerticalJoin([TAi*(ID-TUi),N,(TAi-ID)*TUi]);
    E4:=VerticalJoin([ TUi * TB * TU * TB * TA * TUi * TB *
      TU * TB * TA * TUi * TB * TU +  TUi * TB * TU * TB *
      TA * TUi * TB * TU + TUi*TB*TU, TA * TUi * TB * TU
      * TB *  TA * TUi * TB * TU * TB * TA * TUi * TB * TU
      + TU * TB * TA * TUi * TB * TU * TB * TA * TUi * TB
      * TU + TA * TUi * TB * TU * TB * TA * TUi * TB * TU
      + TU * TB * TA * TUi * TB * TU + TA * TUi * TB * TU
      + TU, -TUi * TB * TU * TB * TA * TUi * TB * TU * TB
      * TA * TUi * TB * TU + (TB * TA * TUi * TB * TU)^2
      - TUi * TB * TU * TB * TA * TUi * TB * TU +
      TB * TA * TUi * TB * TU - TUi*TB*TU + ID ]);

    E:=HorizontalJoin([E1,E2,E3,E4]);

  else
    return "you stop that","space_mat_psl";
  end if;

  return E;
end function;
```

B.1.11 `Space_pgl.m`

```
ToHecke:= recformat <
  der                : ModTupFld,
  inn                : ModTupFld,
  dim                : RngIntElt,
  coeff_dim            : RngIntElt,
  ta                   : Mtrx,
  tb                   : Mtrx,
  tu                   : Mtrx,
  tj                   : Mtrx,
  weight               : SeqEnum,
  level                : RngOrdIdl,
  field                : FldQuad,
  id_index             : RngIntElt,
  d                    : RngIntElt,
  char                 : RngQuadIdl,
  det_twists           : SeqEnum,
  PL                   : SetIndx,
  r                    : UserProgram,
  chi                  : GrpDrchNFElt   >;



// given level ideal J and weight (k,l), we compute the first
// cohomology for PSL(2,Z_K) with coefficients in coinduced-module
// tensor V_(k,l)

DIM:=function(level_data,k,l,a,b)

  level:=level_data`level;
  PA:=level_data`projmatA;
  PAi:=level_data`projmatAi;
  PB:=level_data`projmatB;
  PU:=level_data`projmatU;
  PUi:=level_data`projmatUi;
  PJ:=level_data`projmatJ;
  weight:=[k,l,a,b];
  d:=level_data`d;
```

```
char:=level_data`char;
PL:=level_data`PL;
r:=level_data`r;

K<z>:=QuadraticField(-d);
O:=Integers(K);


A,Ai,B,U,Ui,J:=StandardMats(d);


if char eq 0*O then
  F:=K;
else
  F<t>:=ResidueClassField(char);
end if;


if level ne 1*O then
  TA:=RecursiveMatrix(PA,ModuleMat(char,weight,A));
  TAi:=RecursiveMatrix(PAi,ModuleMat(char,weight,Ai));
  TB:=RecursiveMatrix(PB,ModuleMat(char,weight,B));
  TU:=RecursiveMatrix(PU,ModuleMat(char,weight,U));
  TUi:=RecursiveMatrix(PUi,ModuleMat(char,weight,Ui));
  TJ:=RecursiveMatrix(PJ,ModuleMat(char,weight,J));
else
  TA:=ModuleMat(char,weight,A);
  TAi:=ModuleMat(char,weight,Ai);
  TB:=ModuleMat(char,weight,B);
  TU:=ModuleMat(char,weight,U);
  TUi:=ModuleMat(char,weight,Ui);
  TJ:=ModuleMat(char,weight,J);
end if;
```

```
E:=space_mat_pgl(d,TA,TAi,TB,TU,TUi,TJ);

DER:=Kernel(E);

ID:=TA^0;
F:=HorizontalJoin([ID-TA,ID-TB,ID-TU,ID-TJ]);
INN:=Image(F) meet DER;

dimension:=Dimension(DER)-Dimension(INN);

t,N:=IsPrincipal(level);

Data:=rec< ToHecke |
   der:=DER,
   inn:=INN,
   dim:=dimension,
   coeff_dim:=#Rows(TA),
   ta:=TA,
   tb:=TB,
   tu:=TU,
   tj:=TJ,
   level:=level,
   weight:=[k,l,a,b],
   field:=K,
   id_index:=PLIndex(K,N),
   d:=d,
   char:=char,
   det_twists:=[a,b],
   PL:=PL,
   r:=r,
   chi:=level_data`chi
```

```
  >;

  return Data;
end function;
```

B.1.12 `Utility.m`

```
StandardMats:=function(d)

  K:=QuadraticField(-d);
  O<z>:=MaximalOrder(K);

  A:=Matrix(O,2,2,[1,1,0,1]);
  Ai:=Matrix(O,2,2,[1,-1,0,1]);
  B:=Matrix(O,2,2,[0,-1,1,0]);      /* order 2 */
  U:=Matrix(O,2,2,[1,z,0,1]);
  Ui:=Matrix(O,2,2,[1,-z,0,1]);

  if d in {2,7,11} then
    J:=Matrix(O,2,2,[-1,0,0,1]);
  elif d in {1,3} then
    J:=Matrix(O,2,2,[z,0,0,1]);
  else
    J:="stop that";
  end if;

  return A,Ai,B,U,Ui,J;
end function;


DEWORD:=function(list)
  M:=list[1,1]^0;
  Z:=Integers();
  for i in list do
    M*:=i[1]^Z!i[2];
  end for;
  return M;
end function;


// Write W as a word in [A,i], [B,1], [U, j], [J,k]
WORD:=function(W,d)
  K:=QuadraticField(-d);
  O<z>:=MaximalOrder(K);

  A,Ai,B,U,Ui,J:=StandardMats(d);

  if d eq 1 then

    w:=W;
    S:=[];

    if Determinant(W) eq z then
      Append(~S,[*J,1*]);
      W:=J^(-1)*W;
    elif Determinant(W) eq -z then
      Append(~S,[*J,-1*]);
      W:=J*W;
    elif Determinant(W) eq -1 then
      S cat:= [[*J,1*],[*J,1*]];
      W:=J^2*W;
    end if;

    if Abs(Norm(W[2,1])) ge Abs(Norm(W[1,1])) then
      W:=B*W;
      Append(~S,[*B,1*]);
```

```
end if;

while Norm(W[2,1]) ne 0 do
  q:=O!W[1,1] div O!W[2,1];
  seq:=Eltseq(O!q);
  Append(~S,[*A,seq[1]*]);
  Append(~S,[*U,seq[2]*]);
  Append(~S,[*B,1*]);
  Q:=Matrix(O,2,2,[1,-q,0,1]);
  W:=B*Q*W;
end while;

if W[1,1] eq z then
  S cat:= [[*B,1*],[*U,-1*],[*B,1*],[*U,1*],[*B,1*],[*U,-1*]];
  W:=U^(-1)*B*U*B*U^(-1)*B*W;
elif W[1,1] eq -z then
  S cat:= [[*U,-1*],[*B,1*],[*U,1*],[*B,1*],[*U,-1*],[*B,1*]];
  W:=B*U^(-1)*B*U*B*U^(-1)*W;
end if;

if W[1,1] eq 1 then
  seq:=Eltseq(O!W[1,2]);
  Append(~S,[*A,seq[1]*]);
  Append(~S,[*U,seq[2]*]);
else
  seq:=Eltseq(O!W[1,2]);
  Append(~S,[*A,-seq[1]*]);
  Append(~S,[*U,-seq[2]*]);
end if;

assert (DEWORD(S) eq w or DEWORD(S) eq -w or DEWORD(S) eq z*w
  or DEWORD(S) eq -z*w);
```

```
elif d eq 3 then

 w:=W;

 S:=[];

 while Determinant(W) ne 1 do
   Append(~S,[*J,1*]);
  W:=J^(-1)*W;
 end while;

 if Abs(Norm(W[2,1])) ge Abs(Norm(W[1,1])) then
  W:=B*W;
   Append(~S,[*B,1*]);
 end if;

 while Norm(W[2,1]) ne 0 do
   q:=O!W[1,1] div O!W[2,1];
   seq:=Eltseq(O!q);
   Append(~S,[*A,seq[1]*]);
   Append(~S,[*U,seq[2]*]);
   Append(~S,[*B,1*]);
   Q:=Matrix(O,2,2,[1,-q,0,1]);
  W:=B*Q*W;
 end while;

 R:=Matrix(O,2,2,[z,0,0,z^5]);

 while W[1,1] ne 1 do
   S cat:=[[*B,1*],[*A,1*],[*B,1*],[*U,1*],[*B,1*],
      [*U,-1*],[*B,1*],[*A,-1*]];
  W:=R^(-1)*W;
 end while;
```

```
  seq:=Eltseq(O!W[1,2]);
  Append(~S,[*A,seq[1]*]);
  Append(~S,[*U,seq[2]*]);

  M:=DEWORD(S);
  unitmats:=[z^k*w : k in [1..6]];

  assert M in unitmats;

elif d in {2,7,11} then

  w:=W;
  S:=[];

  if Determinant(W) eq -1 then
    Append(~S,[*J,1*]);
    W:=J*W;
  end if;

  if Abs(Norm(W[2,1])) ge Abs(Norm(W[1,1])) then
    W:=B*W;
    Append(~S,[*B,1*]);
  end if;

  while Norm(W[2,1]) ne 0 do
    q:=O!W[1,1] div O!W[2,1];
    seq:=Eltseq(O!q);
    Append(~S,[*A,seq[1]*]);
    Append(~S,[*U,seq[2]*]);
    Append(~S,[*B,1*]);
    Q:=Matrix(O,2,2,[1,-q,0,1]);
    W:=B*Q*W;
    end while;


    if W[1,1] eq 1 then
      seq:=Eltseq(O!W[1,2]);
      Append(~S,[*A,seq[1]*]);
      Append(~S,[*U,seq[2]*]);
    else
      seq:=Eltseq(O!W[1,2]);
      Append(~S,[*A,-seq[1]*]);
      Append(~S,[*U,-seq[2]*]);
    end if;

    assert (DEWORD(S) eq w or DEWORD(S) eq -w);

  else
    return "you stop that";
  end if;

  return S;
end function;







// identifying ideals via their magma prime factorizations

get_ideal:=function(list,K)
  final:=1*Integers(K);
  for item in list do
    I:=item[1][1]*Integers(K);
    J:=Factorization(I)[item[1][2]][1];
    final:=final*J^item[2];
  end for;
  return final;
end function;


identify_prime:=function(J)
  p:=PrimeDivisors(Norm(J))[1];
  fac:=Factorization(p*Order(J));
  if J eq fac[1][1] then
    return <p,1>;
  else
    return <p,2>;
  end if;
end function;


// factorizes and identifies factors with exponents
identify:=function(I)
  fac:=Factorization(I);
  list:=[];
  for factor in fac do
    Append(~list,<identify_prime(factor[1]),factor[2]>);
  end for;
  return <Norm(I),list>;
end function;


// identifying ideals via their HNF bases
detect_hnf:=function(J,M)
  N:=Norm(J);
  a:=M[1,1];   d:=M[1,2];
  b:=M[2,1];   c:=M[2,2];
  if (d eq 0) and (N eq a*c) and (b in [0..a-1]) then
    return 1;
  else
    return 0;
  end if;
end function;


HNF_basis:=function(J)
  N:=Norm(J);
  M:=BasisMatrix(J);
  Mt:=Matrix(Integers(),2,2,[M[1,2],M[1,1],M[2,2],M[2,1]]);
  HN:=HermiteForm(Mt);
  H:=Matrix(Integers(),2,2,[HN[2,2],HN[2,1],HN[1,2],HN[1,1]]);
  c:=H[2,2];   b:=H[2,1];    a:=H[1,1];
  if c lt 0 then
    c:=-c;    b:=-b;
  end if;

  b:=(b mod a);
  assert 1 eq detect_hnf(J,Matrix(Integers(),2,2,[a,0,b,c]));
  return [Norm(J), b,c];
end function;


get_ideal_hnf:=function(K,list)
  O<w>:=Integers(K);
  N:=list[1];    /* norm */
  gen1:=O!(N/list[3]);
  gen2:=O!(list[2]+w*list[3]);
  return ideal<O| gen1, gen2>;
end function;
```

```
// given a sequence G of algebraic integers a+bw and index r,
// returns the first coordinate "a" of the r'th element of G
R:=function(G,r)
  h:=Eltseq(G[r])[1];
  h:=Integers()!h;
  return h;
end function;


// returns the second coordinate "b"
I:=function(G,r)
  h:=Eltseq(G[r])[2];
  h:=Integers()!h;
  return h;
end function;


// needs to be here for strange compatibility reasons
MatPow:=function(matrix,pow)
  mat_list:=[];
  if pow eq 0 then
    return matrix^0;
  end if;
  for i in [1..Abs(pow)] do
    Append(~mat_list,matrix^Sign(pow));
  end for;
  return &*mat_list;
end function;

// Given a square matrix M and m, it will return 1+M+..+M^(m-1)
// or -M^m*(1+M+...+M^(-m-1)) if k is negative and 0 if k=0
```

```
HeckeMatP:=function(N,M,m)

  W:=N;

   if m lt 0 then
     for j in [0..(-m-1)] do
       W:=W+MatPow(M,j);
     end for;
    W:= -W*MatPow(M,m);
   end if;

   if m gt 0 then
     for j in [0..(m-1)] do
       W:=W+MatPow(M,j);
     end for;
   end if;

   return W;
end function;
```

B.1.13 `Utility_Hecke.m`

```
// This function gathers ONLY RATIONAL eigenvalue systems from a
// given set of commuting matrices.
// needs to be loaded after Hecke functionalities and Artin Algebras
GET_EV_RAT:=function(list)
  A:=Parent(list[1]);
  D:=Decomposition(A);
  EV_List:=<>;

   for j in [1..#D] do
     TT:=[BaseChange(T,D[j]): T in list];
     m:=NumberOfRows(TT[1]);
```

```
    ev_size:=[];
    for M in TT do
      total:=0;
      for item in Eigenvalues(M) do
        total:=total+item[2];
      end for;
    Append(~ev_size,total);
    end for;

    if m eq Min(ev_size) then
      Zx<x>:=PolynomialRing(Integers());
      fx:=Zx!DefiningPolynomial(BaseRing(TT[1]));
      mult:=SetToSequence(Eigenvalues(TT[1]))[1][2];
      ev:=[SetToSequence(Eigenvalues(M))[1][1] : M in TT];
      Append(~EV_List,<fx,mult,ev>);
    end if;
  end for;

  return EV_List;
end function;


// gathers all eigenvalues from a set of commuting matrices in list.
// recursively calls itself and takes large compositums of number fields
// so can be VERY slow
GET_EV:=function(list)
  char:=Characteristic(CoefficientRing(list[1]));
  if char eq 0 then
    F<w>:=OptimizedRepresentation(SplittingField(
      &*[MinimalPolynomial(m) : m in list]));
  else
    F<w>:=SplittingField(&*[MinimalPolynomial(m) : m in list]);
  end if;
```

```
list:=[ChangeRing(h,F) : h in list];
A:=MatrixAlgebra(list);

if Dimension(A) eq 0 then
  EV_list:=[<DefiningPolynomial(F),Ncols(list[1]),
    [0 : i in [1..#list]]>];
  basis:=Basis(Kernel(list[1]));
else
  D:=Decomposition(A);
  EV_list:=<>;

  for j in [1..#D] do
    TT:=[BaseChange(T,D[j]): T in list];
    m:=NumberOfRows(TT[1]);
    ev_size:=[];
    for M in TT do
      total:=0;
      for item in Eigenvalues(M) do
        total:=total+item[2];
      end for;
    Append(~ev_size,total);
    end for;

    if m eq Min(ev_size) then
      Zx<x>:=PolynomialRing(Integers());
      if char eq 0 then
        fx:=Zx!DefiningPolynomial(BaseRing(TT[1]));
      else
        fx:=DefiningPolynomial(BaseRing(TT[1]));
      end if;
      mult:=SetToSequence(Eigenvalues(TT[1]))[1][2];
      ev:=[SetToSequence(Eigenvalues(M))[1][1] : M in TT];
```

```
        Append(~EV_list,<fx,mult,ev>);
      end if;
    end for;
    basis:=&cat[Rows(D[i,1]) : i in [1..#D]];
  end if;

  return EV_list,basis;
end function;



// Takes a list of ideals and returns the relevant Hecke matrices
GetHeckeMatrices:=function(space,list)
  if space`dim ne 0 then // doing hecke stuff
    heckes:=[];
    for TP in list do
      time Append(~heckes,HECKE(TP,space));
    end for;
    return heckes;
  else
    return [];
  end if;
end function;



IsRationalSystem:=function(evs)
  isitrational:=true;
  for e in evs do
    if not e in Integers() then
      isitrational:=false;
    end if;
  end for;
  return isitrational;
end function;
```

B.2 Period polynomials

B.2.1 H2.m

```
// computes H2
H2quo:=function(spec)

  ZK:=MaximalOrder(spec`field);
  z:=spec`field.1;
  d:=spec`d;

  if d eq 1 then

    E:=Matrix(ZK,2,2,[-1,z,z,0]);
    S:=Matrix(ZK,2,2,[0,-1,1,0]);
    SL:=Matrix(ZK,2,2,[0,z,z,0]);
    U:=Matrix(ZK,2,2,[1,-1,1,0]);
    J:=Matrix(ZK,2,2,[z,0,0,1]);

    TE:=Act(spec,E);
    TS:=Act(spec,S);
    TSL:=Act(spec,SL);
    TU:=Act(spec,U);
    TJ:=Act(spec,J);

    KER:=Kernel(1-TE) + Kernel(1-TS) + Kernel(1-TSL) +
       Kernel(1-TU) + Kernel(1+TJ);

  elif d eq 2 then

  A:=Matrix(ZK,2,2,[1,z,z,-1]);
  S:=Matrix(ZK,2,2,[0,-1,1,0]);
```

```
U:=Matrix(ZK,2,2,[1,-1,1,0]);
Tz:=Matrix(ZK,2,2,[1,z,0,1]);
J:=Matrix(ZK,2,2,[-1,0,0,1]);

TA:=Act(spec,A);
TS:=Act(spec,S);
TU:=Act(spec,U);
TTzi:=Act(spec,Tz^-1);
TJ:=Act(spec,J);

KER:=Kernel(1-TA)*(1-TTzi) + Kernel(1-TS) +
   Kernel(1-TU) + Kernel(1+TJ);

 elif d eq 3 then

   LS:=Matrix(ZK,2,2,[0,-z,1-z,0]);
   U:=Matrix(ZK,2,2,[1,-1,1,0]);
   SL:=Matrix(ZK,2,2,[0,-1+z,z,0]);
   J:=Matrix(ZK,2,2,[z,0,0,1]);

   TLS:=Act(spec,LS);
   TU:=Act(spec,U);
   TSL:=Act(spec,SL);
   TJ:=Act(spec,J);

   KER:=Kernel(1-TLS) + Kernel(1-TU) + Kernel(1-TSL) +
      Kernel(1+TJ);

 elif d eq 7 or d eq 11 then

 if d eq 7 then
   A:=Matrix(ZK,2,2,[1,-1+z,z,-1]);
 else
```

```
    A:=Matrix(ZK,2,2,[1,-1+z,z,-2]);
  end if;
    g:=Matrix(ZK,2,2,[0,-1,1,-z]);
    S:=Matrix(ZK,2,2,[0,-1,1,0]);
    U:=Matrix(ZK,2,2,[1,-1,1,0]);
    J:=Matrix(ZK,2,2,[-1,0,0,1]);

    TA:=Act(spec,A);
    TS:=Act(spec,S);
    TU:=Act(spec,U);
    Tg:=Act(spec,g);
    TJ:=Act(spec,J);

    KER:=Kernel(1-TS) + Kernel(1-TU) + Kernel(1-TA)*(1+Tg) +     Kernel(1+TJ);

  end if;

  V:=RSpace(CoefficientRing(KER),Degree(KER));
  H2,m:=quo<V|KER>;

  return H2,m;

end function;


// computes hecke operators on H2
H2Hecke:=function(spec,H2,m,P)
  T:=Heilbronn(P,spec`level);
  H:=[Act(spec,Matrix(spec`field,2,2,t)) : t in T];
  M:= Matrix([m(Inverse(m)(H2.i) * &+H) : i in [1..Dimension(H2)]]);
return M;
end function;
```

B.2.2 `Hecke.m`

```
// Haluk's code for computing Heilbronn matrices
// previously used magma's QuadraticField functionality
Heilbronn:=function(J,level)

  ZK:=Order(level);
  K:=NumberField(ZK);

  // this section translates between number field and quadratic field
  DD:=Discriminant(K);
  Z:=Integers();
  if DD in [-4,-8] then
    K1:=QuadraticField(Z!(DD/4));
  else
    K1:=QuadraticField(Z!DD);
  end if;
  O:=MaximalOrder(K1);

  t,m:=IsIsomorphic(K,K1);

  t,g:=IsPrincipal(J);
  w:=O.2;
  J:=O*m(K!g);

  t,pi:=IsPrincipal(J);
  pi:=O!pi;
  a:=Z!pi[1];
  b:=Z!pi[2];
  q:=Norm(J);

  List:= [**];
```

```
if IsPrime(q) then
  for  s in [0..q-1]  do
    x1 := pi; x2 := -s;  y1 := 0 ; y2 := 1;
    Append(~List, [x1,x2, y1, y2]);

    a:=-pi; b:=s;

    while b ne 0 do
      r:= a mod b;
      q:= O! ((a-r) / b);
      x3:= -x1+q*x2;
      x1 := x2;  x2 := x3;
      y3 := -y1+q*y2;
      y1 := y2;  y2 := y3;

      Append(~List,[ x1, x2, y1, y2]);

      a:=-b; b:=r;
    end while;
  end for;
    Append(~List, [1, 0, 0,  pi]);
    else
    p:=PrimeDivisors(q)[1];
    for  s,t in [0..p-1]  do
      x1 := pi; x2 := -(s+t*w);  y1 := 0 ; y2 := 1;
      Append(~List, [x1,x2, y1, y2]);

      a:=(-pi); b:=(s+t*w);

      while b ne 0 do
        r:= a mod b;
        q:= O! ((a-r) / b);
        x3:= -x1+q*x2;
```

```
          x1 := x2;   x2 := x3;
          y3 := -y1+q*y2;
          y1 := y2;   y2 := y3;

          Append(~List,[ x1, x2, y1, y2]);

          a:=-b; b:=r;
        end while;
      end for;
    Append(~List, [1, 0, 0,  pi]);
  end if;

  // this translates back
  List2:=[];
  mm:=Inverse(m);
  for l in List do
  Append(~List2,[mm(l[1]),mm(l[2]),mm(l[3]),mm(l[4])]);
  end for;

  return List2;
end function;



// Computes the action of the Hecke operator at the prime ideal P.
// returns the operator as it acts on the basis of W given by
// (1 0 ... 0), (0 1 0 ... 0), ... (0 0 ... 1), and the operator
// as it acts on the actual vectors of W as well. Both are useful
// depending on what you want to do with them, so we keep both.
Hecke:=function(W,P)
  T:=Heilbronn(P,W`level);
  H:=[Act(W`spec,Matrix(MaximalOrder(W`field),2,2,
    [t[4],-t[2],-t[3],t[1]])) : t in T];
  HPB:=&+H;
  // forces compatibility, should always come back true
  tt:=IsIsomorphic(CoefficientRing(Parent(HPB)),
    NumberField(Domain(W`down)));

  if W`char ne 0 then
    HPB:=W`down(HPB);
  end if;

  BW:=Basis(W`space);

  HP:=Matrix([Solution(Matrix(BW),b*HPB) : b in BW ]);

  // in case one wants to do things mod p
  if W`char eq 0 then
    F:=BaseRing(W`space);
  else
    F:=CoefficientRing(W`space);
  end if;
  return ChangeRing(HP,F),HPB;
end function;


// Simple iterating function that returns the Hecke matrices attahed
// to W associated to the prime ideals in list
GetHeckeMatrices:=function(W,list)
  HH:=[];
  HHB:=[];

  for P in list do
    time T,TB:=Hecke(W,P);
    Append(~HH,T);
    Append(~HHB,TB);
  end for;

  return HH,HHB;
end function;



// Given a list of commuting matrices (coming in our case from
// Hecke operators), returns the simultaneous eigenvalues of the
// matrices, as well as their field of definition and their
// multiplicities. Uses Wiese's ArtinAlgebras.
// can sometimes take a long time to run, if there are eigenvalues
// in a large degree field. this is partly because it is not written
// with efficiency in mind.
GET_EV:=function(list)

  char:=Characteristic(CoefficientRing(list[1]));

  A:=MatrixAlgebra(list);

  // return the 0 eigenvalue system with maximum multiplicity
  if Dimension(A) eq 0 then
    F:=CoefficientRing(list[1]);
    P<x>:=PolynomialRing(F);
    EV_list:=[<x-1,Ncols(list[1]),[F!0 : i in [1..#list]]>];
    basis:=Basis(Kernel(list[1]));
  else

    D:=Decomposition(A);
    EV_list:=<>;

    if #D ne 1 then
      pieces:=[**];
      for j in [1..#D] do
      TT:=[BaseChange(T,D[j]): T in list];
      if char eq 0 then
        F:=OptimizedRepresentation(SplittingField(
          &*[MinimalPolynomial(u) : u in TT]));
      else
        F<w>:=SplittingField(&*[MinimalPolynomial(u) : u in TT]);
      end if;
      TT:=[ChangeRing(T,F) : T in TT];
      // we re-call this function but with one fewer "piece"
      // this recursiveness is probably not good for the
      // efficiency of the program
      EVs:=$$(TT);
      for e in EVs do
        Append(~pieces,e);
      end for;
    end for;

    return pieces;

  else

    // in the case there is only one eigenvalue system, this
    // part is used
    for j in [1..#D] do
      TT:=[BaseChange(T,D[j]): T in list];
      m:=NumberOfRows(TT[1]);
      ev_size:=[];
      for M in TT do
        total:=0;
        for item in Eigenvalues(M) do
          total:=total+item[2];
        end for;
        Append(~ev_size,total);
        end for;

        if m eq Min(ev_size) then
          Zx<x>:=PolynomialRing(Integers());
          if char eq 0 then
            fx:=Zx!DefiningPolynomial(BaseRing(TT[1]));
          else
            fx:=DefiningPolynomial(BaseRing(TT[1]));
          end if;
          mult:=SetToSequence(Eigenvalues(TT[1]))[1][2];
          ev:=[SetToSequence(Eigenvalues(M))[1][1] : M in TT];
          Append(~EV_list,<fx,mult,ev>);
        end if;
      end for;
      basis:=&cat[Rows(D[i,1]) : i in [1..#D]];
    end if;
  end if;

  return EV_list;
end function;


// Given a list of commuting matrices and vals in the form
// <pol, multiplicity, eigs>, finds a basis of simultaneous
// generalised eigenvectors.
GenEigVecs:=function(W,HH,vals)

  n:=Ncols(HH[1]); //matrices in list are always sqaure and the same size
  d:=vals[2]; // dimension of the space we want
  e:=vals[3];

  FF:=SplittingField(ChangeRing(vals[1],CoefficientRing(W`space)));
  HHe:=[ChangeRing(HH[i],FF) : i in [1..#HH]];
  SP:= [ChangeRing(HH[i],Parent(e[i]))-e[i]*ChangeRing(HH[i]^0,
     Parent(e[i])) : i in [1..#HHe]];

  MM:=[];
  for M in SP do
    M1:=M;
    i:=1;
    while Rank(M1) gt (n-d) or i lt 100 do
      M1*:=M;
      i+:=1; // just in case
    end while;
    Append(~MM,M1);
  end for;

  return Basis(&meet [Kernel(m) : m in MM]);
end function;



// Given Hecke matrices HH and their associated primes HP,
// returns vectors of W, along with their Hecke eigenvalues
// at the primes in HP, and the generators of the primes in HP.
//This uses Wiese's ArtinAlgebras.
  GetPolVals:=function(W,HH,HP)
  KK:=Kernel(ChangeRing(HH[1]-HH[1],CoefficientRing(W`space)));
  h:=hom<W`space -> KK | [KK.i : i in [1..Dimension(KK)]] >;
  g:=Inverse(h); // map from <(1 0 ... 0) ... (0 0 ... 1)> to W

  // first we get the eigenvalues themselves, with no talk
  // of multiplicity
  EVs:=GET_EV(HH);
  EVs_nomult:=[**];
  for e in EVs do
    for i in [1..e[2]] do
    Append(~EVs_nomult,e[3]);
  end for;
end for;

list:=[**];

// now we gather the (generalised) eigenvectors corresponding to
// each EV system
for e in EVs do
  LL:=GenEigVecs(W,HH,e);
  for l in LL do
    Append(~list,l);
  end for;
end for;

//gathers the generators of the hecke operators
gens:=[];
for i in HP do
  t,gen:=IsPrincipal(i);
  Append(~gens,gen);
end for;

// puts everything together
pol_vals:=[];
for i in [1..#list] do
  l:=list[i];
  WW:=ChangeRing(W`space,Parent(EVs_nomult[i,1]));
  ll:=ElementToSequence(l);
  Append(~pol_vals,[*&+[ll[j]*WW.j : j in [1..W`dim]],EVs_nomult[i],gens*]);
end for;

EV_systems:=[**];
for e in EVs do
```

```
    Append(~EV_systems,<e[1],e[2],e[3]>);
  end for;

  return EV_systems,pol_vals;
end function;



// some utility functions for putting ideals into normal form
detect_hnf:=function(J,M)
  N:=Norm(J);
  a:=M[1,1];   d:=M[1,2];
  b:=M[2,1];   c:=M[2,2];
  if (d eq 0) and (N eq a*c) and (b in [0..a-1]) then
    return 1;
  else
    return 0;
  end if;
end function;

HNF_basis:=function(J)
  N:=Norm(J);
  M:=BasisMatrix(J);
  Mt:=Matrix(Integers(),2,2,[M[1,2],M[1,1],M[2,2],M[2,1]]);
  HN:=HermiteForm(Mt);
  H:=Matrix(Integers(),2,2,[HN[2,2],HN[2,1],HN[1,2],HN[1,1]]);
  c:=H[2,2];   b:=H[2,1];    a:=H[1,1];
  if c lt 0 then
    c:=-c;    b:=-b;
  end if;

  b:=(b mod a);
  assert 1 eq detect_hnf(J,Matrix(Integers(),2,2,[a,0,b,c]));
  return [Norm(J), b,c];
```

```
end function;
```

B.2.3 `loading_script_H2.m`

```
AttachSpec("ArtinAlgebras/ArtinAlgebras.spec");

load "ProjAction.m";
load "WeightAction.m";
load "Space.m";
load "Hecke.m";
load "H2.m";

d:=11;

level:=[1,0];
weight:=[10, 10, 0, 0];
HB:=30;
chi:=1;

K:=QuadFld(d);
ZK:=MaximalOrder(K);
level:=(K!level)*ZK;
PL,r:=ProjectiveLine(quo<ZK|level>);
chi:=Elements(DirichletGroup(level))[chi];

SpecData:= recformat <
  d                    : RngIntElt,
  level                : RngOrdIdl,
  weight               : SeqEnum,
  chi                  : GrpDrchNFElt,
  PL                   : SetIndx,
  r                    : UserProgram,
  char_field           : FldNum,
  field                : FldNum >;

spec:=rec<SpecData | d:=d,
  level:=level ,
  weight:=weight ,
  chi:=chi ,
  PL:=PL,
  r:=r ,
  char_field:=Compositum(Codomain(chi),K),
  field:=K >;

print "computing H2...";
time H2,m:=H2quo(spec);

if Dimension(H2) eq 0 then
  print "No EV systems found";
else
  HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level) eq 1*ZK];

  HNF:=[HNF_basis(J): J in HP];
  ParallelSort(~HNF,~HP);

  HH:=[];
  print "finding Hecke matrices...";
  for P in HP do
    time Append(~HH,H2Hecke(spec,H2,m,P));
  end for;
  print "finding eigenvalue systems...";
  EV_systems:=GET_EV(HH);
  print "found the following eigenvalue systems:";
  print EV_systems;
end if;
```

B.2.4 `loading_script_pols.m`

```
AttachSpec("ArtinAlgebras/ArtinAlgebras.spec");

load "ProjAction.m";
load "WeightAction.m";
load "Space.m";
load "Hecke.m";
load "PeriodPols.m";

d:=11;

level:=[1,0];
weight:=[10, 10, 0, 0];
char:=0;
HB:=30;
chi:=1;
type:="GL";

print "computing space of period polynomials...";
time W:=PolSpace(d,level,weight,char,type,chi);

K:=W`field;
ZK:=W`ord;
level:=W`level;

if W`dim eq 0 then
  print "No EV systems found";
else
  if char eq 0 then
    HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level) eq 1*ZK];
  else
    HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level*char) eq 1*ZK];
  end if;
  //HP:=PrimesUpTo(HB,K);
  //HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level) eq 1*ZK];

  HNF:=[HNF_basis(J): J in HP];
  ParallelSort(~HNF,~HP);

  print "computing Hecke matrices...";
  HH,HHB:=GetHeckeMatrices(W,HP);
  print "computing eigenvalue systems and algebraic
    eigen-polynomials...";
  EV_systems,pol_vals:=GetPolVals(W,HH,HP);

  print "found the following eigenvalue systems:";
  EV_systems;
end if;
```

B.2.5 `PeriodPols.m`

```
PeriodPol:=function(W,vec : ind:=W`id_index)
  // if the user wants a different component polynomial, they can ask
  // for it, otherwise we give them the component associated to the
  // identity element. this is the same as the polynomial attached
  // to {0,\infty}
  if not assigned ind then
    ind:=W`id_index;
  end if;
  weight:=W`weight;
  k:=weight[1];
  l:=weight[2];
  R:=CoefficientRing(W`space);

  D:=(k+1)*(l+1);
```

```
   coeffs:=[vec[D*(ind-1)+i] : i in [1..D]];
  M:=Matrix(R,k+1,l+1,coeffs); //turns the coeffs into a k+1-by-l+1 matrix, whic
  return M;
end function;


// Returns the central coefficient(s) of the period polynomial
CentralCoeffs:=function(W,vec)
   weight:=W`weight;
   k:=weight[1];
   l:=weight[2];

   pol,M:=PeriodPol(W,vec);

   k2:=k mod 2;
   l2:=l mod 2;

   i:=(k-k2) div 2 + 1;
   j:=(l-l2) div 2 + 1;

   return Submatrix(M,i,j,k2+1,l2+1);
end function;


// pairs two monomials
PairMon:=function(weight,mon1,mon2)

   PP1:=Parent(mon1);
   PP2:=Parent(mon2);
   degx1:=Degree(mon1,PP1.1);
   degxb1:=Degree(mon1,PP1.3);
  degx2:=Degree(mon2,PP2.1);
  degxb2:=Degree(mon2,PP2.3);

  k:=weight[1];
  l:=weight[2];

  if degx1+degx2 eq k and degxb1+degxb2 eq l then
    return (-1)^(degx1+degxb1) * Binomial(k,degx1)^(-1)
      * Binomial(k,degxb1)^(-1);
  else
    return 0;
  end if;
end function;


// breaks two polynomials up into monomials, pairs them all,
// then adds it all up
PairPol:=function(weight,pol1,pol2)
  mons1:=Monomials(pol1);
  mons2:=Monomials(pol2);

  coeffs1:=Coefficients(pol1);
  coeffs2:=Coefficients(pol2);

  pair:=0;

  for i in [1..#mons1] do
    for j in [1..#mons2] do
      pair+:=coeffs1[i]*coeffs2[j]*PairMon(weight,mons1[i],
        mons2[j]);
    end for;
  end for;
  return pair;
end function;


// input a vector (from pol_vals, or elsewhere) and get a polynomial
VecToPol:=function(W,vec : ind:=W`id_index)

  P<X,Y,Xb,Yb>:=PolynomialRing(CoefficientRing(vec),4);
  c:=ElementToSequence(vec);
  pol:=0;
  k:=W`spec`weight[1];
  l:=W`spec`weight[2];
  for i in [1..k+1] do
    for j in [1..l+1] do
      pol+:=c[(i-1)*k+j+(i-1)]*X^(k-i+1)*Y^(i-1)*
        Xb^(l-j+1)*Yb^(j-1);
    end for;
  end for;
  return pol;
end function;


ScalePol:=function(weight,pol)
  F:=CoefficientRing(Parent(pol));
  ZF:=MaximalOrder(F);
  // here we make sure to not include the first and last terms in
  // the scaling. this needs to work for any polynomial, not just
  // those coming from pol_vals, hence we define the two monomials,
  // then take them away
  PP:=Parent(pol);
  first_mon:=(PP.1*PP.3)^weight[1];
  last_mon:=(PP.2*PP.4)^weight[2];
  pol2:=pol - MonomialCoefficient(pol,first_mon)*first_mon -
    MonomialCoefficient(pol,last_mon)*last_mon;

  coeffs:=Coefficients(pol2);
  ideals :=[ideal<ZF|coeffs[i]> : i in [1..#coeffs]];
  D1:=GCD(ideals[1],ideals[2]);
  for i in [3..#ideals] do
    D1:=GCD(D1,ideals[i]);
  end for;
  return D1;
end function;
```

B.2.6 ProjAction.m

```
// Gives the action of M on the projective line P^1(ZK/J).
ProjMat:=function(J,M)
  ZK:=CoefficientRing(Parent(M));

  if Type(ZK) ne RngOrd then
    ZK:=MaximalOrder(ZK);
  end if;

  SS:=MatrixAlgebra(ZK,2);
  M:=SS!M;

  // this line covers weird magma compatibility problems that
  // can sometimes arise
  t:=IsIsomorphic(NumberField(CoefficientRing(Parent(M))),
    NumberField(Order(J)));

  PL,r:=ProjectiveLine(quo<ZK|J>);

  proj_action:=function(M,i)
```

```
    t,im :=r(PL[i]*M,true,false);
    if t then
      return Index(PL,im);
    else
      return -1; //this picks out the bad elements
    end if;
  end function;

  perm:=[];
  for i in [1..#PL] do
    Append(~perm,proj_action(M,i));
  end for;

  l:=#PL;
  Mat:=[];
  for i in [1..l] do
    new_row:=[];
    for j in [1..l] do
      if j eq perm[i] then
        Append(~new_row,1);
      else
        //we ignore the elements with problems at p, per Wang 94
        // this allows for computing Hecke operators at primes dividing J
        Append(~new_row,0);
      end if;
    end for;
    Append(~Mat,new_row);
  end for;

  return ChangeRing(Matrix(Mat),NumberField(ZK));
end function;
```

```
// This is the usual ProjMat we know and love. Gives the action of M
// on the projective line P^1(ZK/J). We supply the projective line and
// its decision function r because Magma sometimes generates different
// orderings of elements in the PL

ProjMatChi:=function(spec,M)

  ZK:=CoefficientRing(Parent(M));

  if Type(ZK) ne RngOrd then
    ZK:=MaximalOrder(ZK);
  end if;

  SS:=MatrixAlgebra(ZK,2);
  M:=SS!M;

  PL:=spec`PL;
  r:=spec`r;

  proj_action:=function(M,i)
    t,im,scal :=r(PL[i]*M,true,true);
    if t then
      return Index(PL,im),scal;
    else
      return -1,0; //this picks out the bad elements
    end if;
  end function;

  perm:=[];
  scalars:=[];
  for i in [1..#PL] do
    pa,sc:=proj_action(M,i);
    Append(~perm,pa);
    Append(~scalars,sc);
  end for;

  l:=#PL;
  Mat:=[];
  for i in [1..l] do
    new_row:=[];
    for j in [1..l] do
      if j eq perm[i] then
        Append(~new_row,spec`field!spec`chi(scalars[i]));
      else
        //we ignore the elements with problems at p, per Wang 94
        Append(~new_row,spec`field!0);
      end if;
    end for;
    Append(~Mat,new_row);
  end for;

  return ChangeRing(Matrix(Mat),spec`field);
end function;


// A simple function that returns the index of (0 : 1) so we know
// where to look for our period polynomial
IdIndex:=function(level)
  ZK:=Ring(Parent(level));
  PL,r:=ProjectiveLine(quo<ZK|level>);
  return Index(PL,Vector([ZK!0,ZK!1]));
end function;


// for compatibility reasons, we use the same fields that the LMFDB uses
QuadFld:=function(d)
```

```
  _<x>:=PolynomialRing(Integers());
  if d in [1,2] then
    return NumberField(x^2+d);
  elif d in [3,7,11] then
    return NumberField(x^2-x-Integers()!((-d-1)/4));
  else
    return "d = " cat Sprint(d) cat " currently not supported";
  end if;
end function;
```

B.2.7 Space.m

```
// Combines the action on the projective line and the action on
// the polynomial space V_{k,l}(C). Covers both trivial cases.
Act:=function(spec,mat)
  PM:=ProjMatChi(spec,mat);
  WM:=WeightMat(spec,mat);

  return TensorProduct(PM,WM);
end function;


// Handles setting up the various spaces for d in {1,2,3,7,11}
StandardMats:=function(spec)

  ZK:=CoefficientRing(spec`PL[1]);
  K<z>:=NumberField(ZK);

  d:=spec`d;

  if d eq 1 then

    T:=Matrix(ZK,2,2,[1,1,0,1]);
    S:=Matrix(ZK,2,2,[0,-1,1,0]);
    Tz:=Matrix(ZK,2,2,[1,z,0,1]);
    L:=Matrix(ZK,2,2,[z,0,0,-z]);
    J:=Matrix(ZK,2,2,[z,0,0,1]);

    TT:=Act(spec,T);
    TS:=Act(spec,S);
    TTz:=Act(spec,Tz);
    TL:=Act(spec,L);
    TJ:=Act(spec,J);
    ID:=TT^0;

    TE:=Act(spec,Tz*S*L);

    return [
    ID+TS, ID-TL, ID+TT*TS+(TT*TS)^2, ID+TE+TE^2, ID-TJ
    ];

  elif d eq 2 then

    T:=Matrix(ZK,2,2,[1,1,0,1]);
    S:=Matrix(ZK,2,2,[0,-1,1,0]);
    Tz:=Matrix(ZK,2,2,[1,z,0,1]);
    Tzi:=Matrix(ZK,2,2,[1,-z,0,1]);
    J:=Matrix(ZK,2,2,[-1,0,0,1]);

    TT:=Act(spec,T);
    TS:=Act(spec,S);
    TTz:=Act(spec,Tz);
    TTzi:=Act(spec,Tzi);
    TJ:=Act(spec,J);
    ID:=TT^0;
    TTzS:=TTz*TS; //a little bit of time saving

    return [
    ID+TS, ID+TT*TS+(TT*TS)^2, ID + TS*TTz + TTzS + TTzi*TS*TTzS,
      ID-TJ
    ];

  elif d eq 3 then

    T:=Matrix(ZK,2,2,[1,1,0,1]);
    S:=Matrix(ZK,2,2,[0,-1,1,0]);
    Tz:=Matrix(ZK,2,2,[1,z,0,1]);
    L:=Matrix(ZK,2,2,[z,0,0,-z^2]);
    J:=Matrix(ZK,2,2,[z,0,0,1]);

    TT:=Act(spec,T);
    TS:=Act(spec,S);
    TTz:=Act(spec,Tz);
    TL:=Act(spec,L);
    TJ:=Act(spec,J);
    ID:=TT^0;

    TE:=Act(spec,T^-1*Tz*S*L);

    return [
    ID+TS, ID-TL, ID+TT*TS+(TT*TS)^2, ID+TE+TE^2, ID-TJ
    ];

  elif d eq 7 then

    T:=Matrix(ZK,2,2,[1,1,0,1]);
    S:=Matrix(ZK,2,2,[0,-1,1,0]);
    Tz:=Matrix(ZK,2,2,[1,z,0,1]);
    J:=Matrix(ZK,2,2,[-1,0,0,1]);

    TT:=Act(spec,T);
    TS:=Act(spec,S);
    TTz:=Act(spec,Tz);
    TJ:=Act(spec,J);
    ID:=TT^0;

    return [
    ID+TS, ID+TT*TS+(TT*TS)^2, TT+TS*TTz+TTz*TS*TT+TS*TTz^-1*TS*TTz,
      ID-TJ
    ];

  elif d eq 11 then

  T:=Matrix(ZK,2,2,[1,1,0,1]);
  S:=Matrix(ZK,2,2,[0,-1,1,0]);
  Tz:=Matrix(ZK,2,2,[1,z,0,1]);
  J:=Matrix(ZK,2,2,[-1,0,0,1]);
  E:=Tz^-1*S*Tz*S*T;
  F:=Matrix(ZK,2,2,[-1,0,0,-1]);

  TT:=Act(spec,T);
  TS:=Act(spec,S);
  TTz:=Act(spec,Tz);
  TE:=Act(spec,E);
  TEi:=Act(spec,E^-1);
  TJ:=Act(spec,J);
  ID:=TT^0;

  return [
  ID+TS, ID+TT*TS+(TT*TS)^2, TT + TS*TTz + TT*TE + TS*TTz*TEi
    + TTz*TS*TT+TS*TTz^-1*TS*TTz, ID-TJ
  ];


  else
    print "bad choice for Q(sqrt(-d))";
    return -1;
  end if;
end function;


// Defines the space of polynomials invariant under the action of
// Gamma_0(level). Uses the matrices in Sengun ExpMath.
PolSpace:=function(d,level,weight,char,type,chi)

  // this will be the final format we output the modular form space in
  PolData:= recformat <
    space              : ModTupFld,
    level              : RngOrdIdl,
    chi                : GrpDrchNFElt,
    weight             : SeqEnum,
    dim                : RngIntElt,
    id_index           : RngIntElt,
    d                  : RngIntElt,
    field              : FldNum,
    ord                : RngOrd,
    char               : RngIntElt,
    down               : Map,
    spec               : Rec >;

  // this format type allows us to package up all the data specifying
  // which space we care about: d for the field, level, weight,
  // character, projective line, and the field all of the matrices
  // are going to live in
```

```
SpecData:= recformat <
  d                : RngIntElt,
  level            : RngOrdIdl,
  weight           : SeqEnum,
  chi              : GrpDrchNFElt,
  PL               : SetIndx,
  r                : UserProgram,
  field            : FldNum >;

K<z>:=QuadFld(d);
ZK:=MaximalOrder(K);
level:=ZK!level;

PL,r:=ProjectiveLine(quo<ZK|level>);
// for compatibility reasons these lines needs to be here
// otherwise ProjActionChi can't evaluate chi at the
// scalar given by r.
level:=Parent(1*CoefficientRing(PL[1]))!level;
ZK:=Order(level);
K:=NumberField(ZK);


DG:=Elements(DirichletGroup(level));
chi:=DG[chi];

spec:=rec<SpecData | d:=d,
level:=level,
weight:=weight,
chi:=chi,
PL:=PL,
r:=r,
field:=Compositum(Codomain(chi),K) >;
```

```
M:=StandardMats(spec);

 // in characteristic p we want to send all of our matrices that
 // give the action to the residue class field of a prime over p*ZK.
 // it doesn't really matter which
 if char ne 0 then
   KK:=Compositum(BaseRing(M[1]),Codomain(chi));
   ZKK:=MaximalOrder(KK);
   F,down:=ResidueClassField(Factorization(char*ZKK)[1,1]);
 else
   down:=IdentityHomomorphism(Parent(M[1]));
 end if;

 // we always put the [e,0,0,1] relation at the end of the list
 // so we can do SL calculations if we want by just omitting the
 // last relation. here e generates the unit group of ZK
 if type eq "GL" then
   W:=&meet [Kernel(down(u)) : u in M];
 elif type eq "SL" then
   W:=&meet [Kernel(down(u)) : u in M[1..#M-1]];
 end if;

W:=rec<PolData | space:=W,
   level:=level,
   chi:=chi,
   weight:=weight,
   dim:=Dimension(W),
   id_index:=IdIndex(level),
   d:=d,
   field:=K,
   ord:=ZK,
   char:=char,
   down:=down,
    spec:=spec
  >;

  return W;
end function;
```

B.2.8 `WeightAction.m`

```
// Computes the action of mat on the weight module. Variable weight
// is a 4 entry list [k,l,a,b], where k and l are the degrees of the
// polynomials and a and b are the twists on the two spaces V_k, V_l
// respectively.
WeightMat:=function(spec,mat);

  weight:=spec`weight;

  R:=CoefficientRing(Parent(mat));
  K:=NumberField(R);

  P<x,y>:=PolynomialRing(spec`field,2);

  Symm:=function(k,d,T)
    ST:=ZeroMatrix(spec`field,k+1,k+1);
    for i in [0..k] do
      Q:=(T[1,1]*x+T[1,2]*y)^(k-i)*(T[2,1]*x+T[2,2]*y)^(i);
      for j in [0..k] do
        ST[i+1,j+1]:=MonomialCoefficient(Q,x^(k-j)*y^(j));
      end for;
    end for;
    return Determinant(T)^d*ST;
  end function;

  k:=weight[1];
```

```
   l:=weight[2];
   a:=weight[3];
   b:=weight[4];

   con:=Automorphisms(K)[2];

   matc:=Matrix(R,2,2,[R!con(mat[1][1]),R!con(mat[1][2]),
      R!con(mat[2][1]),R!con(mat[2][2])]);
  TM:=TensorProduct(Symm(k,a,mat),Symm(l,b,matc));

   return TM;
end function;
```

B.2.9 `example.m`

```
AttachSpec("ArtinAlgebras/ArtinAlgebras.spec");

load "ProjAction.m";
load "WeightAction.m";
load "Space.m";
load "Hecke.m";
load "PeriodPols.m";
load "H2.m";

d:=11;

level:=[1,0];
weight:=[10,10, 0, 0 ];
char:=0;
HB:=30;
chi:=1;
type:="GL";
```

```
time W:=PolSpace(d,level,weight,char,type,chi);

K:=W`field;
ZK:=W`ord;
level:=W`level;

if W`dim eq 0 then
  print "No EV systems found";
else
  if char eq 0 then
    HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level) eq 1*ZK];
  else
    HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level*char) eq 1*ZK];
  end if;
  //HP:=PrimesUpTo(HB,K);
  //HP:=[TP : TP in PrimesUpTo(HB,K) | GCD(TP,level) eq 1*ZK];

  HNF:=[HNF_basis(J): J in HP];
  ParallelSort(~HNF,~HP);

  print "finding Hecke matrices...";
  HH,HHB:=GetHeckeMatrices(W,HP);
  print "finding eigenvalue systems...";
  EV_systems,pol_vals:=GetPolVals(W,HH,HP);

  EV_systems;
end if;

// now we do computations with the period polynomials
F:=NumberField(EV_systems[Index([Degree(u[1]) : u in EV_systems],4)][1]);
ZF:=MaximalOrder(F);

// the period polynomial of delta
```

```
r_Delta:=VecToPol(W,pol_vals[Index([u[3][1] : u in EV_systems],252)][1]);

irr_pols:=[ pol_vals[i] : i in [1..W`dim] | IsIsomorphic(
  CoefficientRing(pol_vals[i][1]),F) ];
// the period polynomials of F1 and F2
r_F1:=VecToPol(W,irr_pols[Index([u[3][1] : u in EV_systems |
  IsIsomorphic(Parent(u[3][1]),F)],2*F.1^2 + 3375)][1]);
r_F2:=VecToPol(W,irr_pols[Index([u[3][1] : u in EV_systems |
  IsIsomorphic(Parent(u[3][1]),F)],-2*F.1^2 - 4075)][1]);

// creates the H2 space
H2,m:=H2quo(W`spec);
// we only need one hecke operator to split up the space
HH3:=H2Hecke(W`spec,H2,m,HP[1]);

// now we collect the various corresponding polynomials from H2
v_Delta:=VecToPol(W,Inverse(m)(Kernel(HH3-252).1));

// for compatibility reasons, Inverse(m) won't accept a vector
// defined over F
// so we can cheat and do its job for it
v1:=Kernel(HH3-(2*F.1^2 + 3375)).1;
v_F1:=VecToPol(W,&+[v1[i]*ChangeRing(Inverse(m)(H2.i),F) : i in [1..4]]);

v2:=Kernel(HH3-(-2*F.1^2 - 4075)).1;
v_F2:=VecToPol(W,&+[v2[i]*ChangeRing(Inverse(m)(H2.i),F) : i in [1..4]]);

// first we do Delta
tt,gg:=IsPrincipal(ScalePol(weight,r_Delta)*ScalePol(weight,v_Delta));
D:=Rationals()!gg;

DeltaPair:=PairPol(weight,r_Delta,v_Delta) / D;
print "Pairing of r_Delta and v_Delta:", DeltaPair;
print "Factorization of numerator:",
  Factorization(Integers()!Numerator(DeltaPair));

// next we do F1
F1Pair:=PairPol(weight,r_F1,v_F1)*ZF / (ScalePol(weight,r_F1) *
   ScalePol(weight,v_F1));
print "(Norm of numerator of) Pairing of r_F1 and v_F1:",
  Factorization(Integers()!Numerator(Norm(F1Pair)));

// and F2, which is essentially the same
F2Pair:=PairPol(weight,r_F2,v_F2)*ZF / (ScalePol(weight,r_F2)
  * ScalePol(weight,v_F2));
print "(Norm of numerator of) Pairing of r_F2 and v_F2:",
  Factorization(Integers()!Numerator(Norm(F2Pair)));

// for the eiseinstein-genuine-cusp congruences, we note the following:
print "(Norm of) Denominator of leading coeff of Delta:", Factorization(
  Numerator(Norm(ScalePol(weight,r_Delta))));
print "(Norm of) Denominator of leading coeff of F1:", Factorization(
  Numerator(Norm(ScalePol(weight,r_F1))));
```

BIBLIOGRAPHY

[AS86] Avner Ash and Glenn Stevens. Cohomology of arithmetic groups and congruences between systems of hecke eigenvalues. *Journal fÃĳr die reine und angewandte Mathematik*, 365:192–220, 1986.

[Ash94] A. Ash. Unstable cohomology of sl(n,o). *Journal of Algebra*, 167(2):330âĂŞ342, 1994.

[BCP97] WIEB BOSMA, JOHN CANNON, and CATHERINE PLAYOUST. The magma algebra system i: The user language. *Journal of Symbolic Computation*, 24(3âĂŞ4):235âĂŞ265, 1997.

[Bel09] Joel Bellaiche. An introduction to the conjecture of bloch and kato, 2009.

[BK07] Spencer Bloch and Kazuya Kato. L-functions and tamagawa numbers of motives. *The Grothendieck Festschrift*, page 333âĂŞ400, 2007.

[Bro12] Kenneth S. Brown. *Cohomology of groups*. Springer, 2012.

[BS73] A. Borel and J-P. Serre. Corners and arithmetic groups. *Commentarii Mathematici Helvetici*, 48(1):436âĂŞ491, 1973.

[Cas62] J. W. S. Cassels. Arithmetic on curves of genus 1 : Iii. the tate-ÅăafareviÄŊ and selmer groups. *Proceedings of the London Mathematical Society*, s3-12(1):259–296, 1962.

[Coh12] Henri Cohen. *Advanced topics in computational number theory*. Springer, 2012.

[Com23] Lewis Combes. Bianchi period polynomials: Hecke action and congruences. 2023.

[Cre84] J. E. Cremona. Hyperbolic tessellations, modular symbols, and elliptic curves over complex quadratic fields. *Compositio Mathematica*, 51(3):275–324, 1984.

[Cre97] J. E. Cremona. *Algorithms for modular elliptic curves.* Cambridge University Press, 1997.

[Ş08] Mehmet Haluk Şengün. *SerreâĂŹs conjecture over imaginary quadratic fields.* PhD thesis, University of Wisconsin-Madison, 2008.

[Ş11] Mehmet Haluk Şengün. On the integral cohomology of bianchi groups. *Exp. Math.*, 20(4):487–505, 2011.

[CV12] Frank Calegari and Akshay Venkatesh. A torsion jacquet–langlands correspondence, 2012.

[CW94] J. E. Cremona and E. Whitley. Periods of cusp forms and elliptic curves over imaginary quadratic fields. *Mathematics of Computation*, 62(205):407âĂŞ429, 1994.

[DS16] Fred Diamond and Jerry Michael Shurman. *A first course in modular forms.* Springer, 2016.

[DSW04] Neil Dummigan, William Stein, and Mark Watkins. Constructing elements in shafarevichâĂŞtate groups of modular motives. *Number Theory and Algebraic Geometry*, page 91âĂŞ118, 2004.

[EPW05] Matthew Emerton, Robert Pollack, and Tom Weston. Variation of iwasawa invariants in hida families. *Inventiones mathematicae*, 163(3):523–580, 2005.

[FGT10] Tobias Finis, Fritz Grunewald, and Paulo Tirao. The cohomology of lattices in sl(2, c). *Experimental Mathematics*, 19(1):29âĂŞ63, 2010.

[Fin89] Benjamin Fine. *Algebraic theory of the Bianchi groups.* Marcel Dekker, 1989.

[Gha99] Eknath Ghate. Critical values of the twisted tensor l-function in the imaginary quadratic case. *Duke Mathematical Journal*, 96(3):595–638, 1999.

[Hab83] Klaus Haberland. Perioden von modulformen einer variabler und gruppencohomologie, i. *Mathematische Nachrichten*, 112(1):245âĂŞ282, 1983.

[Har75] G. Harder. *On the cohomology of SL(2,O)*, page 139âĂŞ150. Adam Hilger, 1975.

[Har20] David Harari. *Galois cohomology and class field theory.* edp Sciences, 2020.

[Hid94] Haruzo Hida. On the critical values of l-functions of gl(2) and gl(2)ÃŮgl(2). *Duke Mathematical Journal*, 74(2), 1994.

[Kar22] Cihan Karabulut. From binary hermitian forms to parabolic cocycles of euclidean bianchi groups. *Journal of Number Theory*, 236:71–115, 2022.

[Koc02] Helmut Koch. *Galois theory of p-extensions.* Springer Monographs in Mathematics. Springer, 1 edition, 2002.

[KZ84] Winfried Kohnen and Don Zagier. Modular forms with rational periods. *Modular Forms (ed. R A Rankin), Ellis Horwood, Chichechester*, page 197âĂŞ249, 1984.

[Lan14] Serge Lang. *Algebraic number theory.* Springer, 2014.

[LMF23] The LMFDB Collaboration. The L-functions and modular forms database. `https://www.lmfdb.org`, 2023. [Online; accessed 14 September 2023].

[Man73] Ju I Manin. Periods of parabolic forms and p-adic hecke series. *Mathematics of the USSR-Sbornik*, 21(3):371âĂŞ393, 1973.

[Moh14] Adam Mohamed. Universal hecke l-series associated with cuspidal eigenforms over imaginary quadratic fields. In Gebhard Böckle and Gabor Wiese, editors, *Computations with Modular Forms*, pages 225–256, Cham, 2014. Springer International Publishing.

[PP13] Vicentiu Pasol and Alexandru A. Popa. Modular forms and period polynomials. *Proceedings of the London Mathematical Society*, 107(4):713–743, 02 2013.

[Rc13] Alexander D. Rahm and Mehmet Haluk Şengün. On level one cuspidal bianchi modular forms. *LMS Journal of Computation and Mathematics*, 16:187âĂŞ199, 2013.

[Rub00] Karl Rubin. *Euler Systems. (AM-147)*. Princeton University Press, 2000.

[Sch15] Peter Scholze. On torsion in the cohomology of locally symmetric varieties. *Annals of Mathematics*, 182:945âĂŞ1066, 2015.

[Ser87] Jean-Pierre Serre. Sur les representations modulaires de degre 2 de Gal($\overline{\mathbb{Q}}/\mathfrak{q}$). *Duke Mathematical Journal*, (54):1279–230, 1987.

[Sil09] Joseph H. Silverman. *The arithmetic of elliptic curves*. Springer New York, 2009.

[Sut16] Andrew V. Sutherland. Computing images of galois representations attached to elliptic curves. *Forum of Mathematics, Sigma*, 4:e4, 2016.

[SV83] Joachim Schwermer and Karen Vogtmann. The integral homology of sl2 and psl2 of euclidean imaginary quadratic integers. *Commentarii mathematici Helvetici*, 58:573–598, 1983.

[Tor12] Rebecca Torrey. On serre's conjecture over imaginary quadratic fields. *Journal of Number Theory*, 132(4):637âĂŞ656, 2012.

[Urb95] Eric Urban. Formes automorphes cuspidales pour gl_2 sur un corps quadratique imaginaire. valeurs spÃľciales de fonctions l et congruences. *Compositio Mathematica*, 99(3):283–324, 1995.

[Was96] Lawrence C. Washington. *Introduction to cyclotomic fields*. Springer, 1996.

[Wei94] Charles A. Weibel. *An introduction to homological algebra.* Cambridge university press, 1994.

[Wil17] Chris Williams. P-adic l-functions of bianchi modular forms. *Proceedings of the London Mathematical Society*, 114(4):614âĂŞ656, 2017.