THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCE

ALGORITHMS FOR COMPUTING CONGRUENCES BETWEEN MODULAR FORMS

By

RANDY HEATON

A Dissertation submitted to the
Department of Mathematics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Summer Semester, 2012

Randy Heaton defended this dissertation on June 28, 2012.

The members of the supervisory committee were:

Amod Agashe
Director

Mark van Hoeij
Co-Directing Dissertation

Simon Capstick
University Representative

Ettore Aldrovandi
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with the university requirements.

ACKNOWLEDGMENTS

In matters that are non-mathematical, I'm gratefull to my parents who motivated me and supported me financially for the early part of my education. My ex-wife Michelle created a home environment that allowed me to focus my efforts during the hardest part of my graduate career; during this time, she sustained me with her love and also supported us both during the months before I began my TA-ship. I'm grateful to Dr. Bettye Anne Case for her guidance, personal and professional, and for the detail with which she took to the task of improving the lives of graduate students. I'm especially grateful for the friendships cultivated in Lov 251, and later elsewhere. They have made my life full and opened me to immensely rewarding perspectives, almost none of which have been mathematical (directly).

Amod Agashe advised me since my second year in graduate school and patiently worked with me through my thesis. His direction and advice have been critical in shaping my understanding of number theory. Mark van Hoeij co-advised me through the later part of my thesis work. During this time, I learned from him how to be a better problem solver. Many of the techniques and ideas in this thesis were taken from generous descriptions given to me by these advisors. In addition, Ettore Aldrovandi has been for several years a resource on varying matters and a patient teacher.

I've had a number of correspondances that have informed my understanding of mathematics and helped me execute the writing of this thesis. To give an incomplete list, conversations with William Stein, Jeremy Rouse, Ernie Croot, Mike Burmester, Craig Citro, Eko Hiranaka, and David Valdivia have been very fruitfull.

TABLE OF CONTENTS

List of Tables . . . vi

Abstract . . . vii

1 Introduction: Modular Forms, Primes of Congruence, and the Birch and Swinnerton-Dyer Conjecture **1**

1.1 Modular Forms . . . 1

1.1.1 Hecke Operators and the structure of $S_2(\Gamma_0(N))$ . . . 2

1.1.2 Computing with Modular Forms . . . 3

1.1.3 Modular Elliptic Curves . . . 6

1.1.4 Congruence Primes . . . 7

1.2 The Birch and Swinnerton-Dyer Conjecture . . . 7

1.3 Motivation for this Thesis: Cancellations in the Conjectural Birch and Swinnerton-Dyer Formula . . . 9

2 Intersection Numbers Between Spaces of Cuspidal Modular Forms **11**

2.1 Introduction . . . 11

2.2 Proof of Theorem 1 . . . 12

2.3 The Algorithm . . . 15

2.3.1 Computing a Basis for $\mathbf{T} \otimes \mathbb{Q}$ . . . 15

2.3.2 Solving the Linear System . . . 16

3 Computing a q-Expansion Basis for $S_2(\Gamma_0(N))$ to high precision **17**

3.1 Introduction . . . 17

3.2 Matrices for Operators on Invariant Subspaces . . . 19

3.2.1 Example: Matrices for Hecke Operators on the New Subspace when the Old and Eisenstein Subspaces Dominate . . . 19

3.2.2 Restricting to Irreducible $\mathbf{T}$-Invariant Subspaces G_f . . . 20

3.2.3 The Algorithm . . . 22

3.3 Local Saturation . . . 24

3.3.1 Computing Integral Bases of q-Expansions . . . 27

3.4 Applications . . . 27

3.4.1 Computing Congruence Primes . . . 27

3.4.2 Cancellations in the Conjectural Birch and Swinnerton-Dyer Formula . . . 29

.1 Finding an element of $\mathbf{T}_{|\mathbf{N}}$ with distinct eigenvalues . . . 29

.2 Computing a multiplicative bound Ω for $[sat(M) : M]$ . . . 30

.3 Code 30
.3.1 Algorithms 1 and 2 30
.3.2 Algorithms 3 and 4 33
.3.3 Algorithm 8 43
References 44
Biographical Sketch 46

LIST OF TABLES

3.1 Tests at composite levels. 23

3.2 Tests at prime levels. 24

3.3 Tests at levels where newforms with integer coefficients dominate the new subspace and we compute a large portion of coefficients by counting points on the associated elliptic curves mod p. 24

3.4 Saturating a $\mathbb{Q}$ basis for $S_2(\Gamma_0(N),\mathbb{Q})$ to compute a basis for $S_2(\Gamma_0(N),\mathbb{Z})$ at levels 500 and 1000 at varying precisions. 26

3.5 Saturating $\mathbb{Q}$ bases for $S_2(\Gamma_0(N),\mathbb{Q})$ computed with SAGE's saturation() and using Algorithm 6 27

3.6 Computing an integral basis for $S_2(\Gamma_0(N))$ 27

ABSTRACT

Let N be a positive integer. We first discuss a method for computing intersection numbers between subspaces of $S_2(\Gamma_0(N), \mathbb{C})$. Then we present a new method for computing a basis of q-expansions for $S_2(\Gamma_0(N), \mathbb{Q})$, describe an algorithm for saturating such a basis in $S_2(\Gamma_0(N), \mathbb{Z})$, and show how these results have applications to computing congruence primes and studying cancellations in the conjectural Birch and Swinnerton-Dyer formula.

CHAPTER 1

INTRODUCTION: MODULAR FORMS, PRIMES OF CONGRUENCE, AND THE BIRCH AND SWINNERTON-DYER CONJECTURE

In this chapter, we define modular forms and congruence primes and give a description of the conjectural Birch and Swinnerton-Dyer Formula. The material will serve as background for chapters one and two of this thesis. We close the chapter with a description of the motivation for this thesis in terms of the BSD conjecture and an overview of chapters two and three.

1.1 Modular Forms

In this section, we present some background information about modular forms. For more information, see [6] or [7].

Throughout this section, we take Γ to be a (multiplicative) subgroup of $SL_2(\mathbb{Z})$. When N is a positive integer, we denote by $\Gamma_0(N)$ the subgroup of $SL_2(\mathbb{Z})$ of matrices of that reduce modulo N to a matrix of the form $\begin{pmatrix} a & b \\ 0 & d \end{pmatrix}$. The integer N will also be referred to as the *level* of $\Gamma_0(N)$. The group Γ acts on the complex upper half plane $\mathfrak{H}$ by linear fractional transformations.

Definition 1. *Let k be a positive integer. A meromorphic function f on $\mathfrak{H}$ is said to be* ***weakly modular of weight** k **for*** Γ *if it satisfies $f(\frac{az+b}{cz+d}) = (cz+d)^k f(z)$ for all $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ in Γ.*

Because $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ lies in $\Gamma_0(N)$ and acts as the translation $z \mapsto z+1$, any weakly modular function for $\Gamma_0(N)$ is periodic with period 1. So there exists a function g such that $f(z) = g(q)$ where $q = e^{2\pi iz}$. We say that f is *holomorphic at* ∞ if the associated function g is holomorphic at $q = 0$. Such an f will have a Fourier expansion $f(z) = g(q) = \sum_{n=0}^{\infty} a_n(f)q^n$.

The orbits of $\mathbb{Q} \cup \infty$ under the action of Γ are called *cusps* for Γ. Let r be a cusp. Because any two elements of $\mathbb{Q} \cup \infty$ are $SL_2(\mathbb{Z})$-equivalent, we can find $\alpha_r = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z})$ such that $\alpha_r(\infty) = r$. Then we say that f is *holomorphic at the cusp r* if $(cz+d)^{-k} f(\alpha_r(z))$ is holomorphic at ∞.

Definition 2. *Let k be a positive integer. A **modular form of weight k** for Γ is a holomorphic function f on $\mathfrak{H}$ that satisfies:*

1. *f is weakly modular of weight k for Γ.*
2. *f is holomorphic at the cusps for Γ.*

A *cusp form* is a modular form f that satisfies the additional condition that the constant coefficient $a_0(f)$ in the Fourier expansion of f is zero. The set of cusp forms of weight k for the group Γ is denoted $S_k(\Gamma)$.

We denote by $Y_0(N)$ the Riemann surface $\mathfrak{H}/\Gamma_0(N)$. By adjoining to $Y_0(N)$ the set of cusps for $\Gamma_0(N)$, we get a compact Riemann surface (i.e. an algebraic curve over $\mathbb{C}$) called the *modular curve* for $\Gamma_0(N)$ and denoted $X_0(N)$.

Theorem 1. *$S_2(\Gamma_0(N))$ is a vector space over $\mathbb{C}$ whose dimension is the genus of $X_0(N)$.*

Proof. See [6] Corollary 1.13. □

1.1.1 Hecke Operators and the structure of $S_2(\Gamma_0(N))$

In this section we introduce *Hecke operators*, which act on $S_2(\Gamma_0(N))$, and describe the structure of $S_2(\Gamma_0(N))$ with respect to this action. To this end, we start by defining an inner product on $S_2(\Gamma_0(N))$.

Definition 3. *Let f and g be two cusp forms in $S_2(\Gamma_0(N))$. We define the Petersson inner product to be*

$$\langle f,g\rangle = \frac{i}{8\pi^2}\int_{X_0(N)} f(z)\overline{g(z)}\,dz \tag{1.1.1}$$

For $n \in \mathbb{Z}$, we define the Hecke operator T_n as follows:

Definition 4. *If p is prime and $(p,N)=1$, the **Hecke operator** T_p is defined by*

$$T_p(f(z)) = \frac{1}{p}\sum_{i}^{p-1} f(\frac{z+i}{p}) + pf(pz) \tag{1.1.2}$$

If p is prime and $(p,N)>1$, then

$$T_p(f(z) = \frac{1}{p}\sum_{i=0}^{p-1} f(\frac{z+1}{p}). \tag{1.1.3}$$

For $n>0$,

$$T_{p^{n+1}} = T_pT_{p^n} - \langle p\rangle T_{p^{n-1}} \tag{1.1.4}$$

If $n=\prod_i p_i^{e_i}$ gives a prime factorization of n, then we set

$$T_n = \prod_i T_{p_i^{e_i}}. \tag{1.1.5}$$

We define the *Hecke Algebra* of $S_2(\Gamma_0(N))$ as $\mathbb{Z}[T_1, T_2, T_3, \ldots]$ and denote it by $\mathbf{T}$.

Let M and N be two positive integers with $M|N$. If $d|\frac{N}{M}$ and $f \in S_2(\Gamma_0(M))$, it is easy to check that $f(dz) \in S_2(\Gamma_0(N))$. The map $\beta_d : S_2(\Gamma_0(M)) \to S_2(\Gamma_0(N))$ defined by $\beta_d : f(z) \mapsto f(dz)$ is called the *d-degeneracy map* from level M. On Fourier coefficients, we have

$$\beta_d : \sum_{i=1}^{\infty} a_i(f)q^i \mapsto \sum_{i=1}^{\infty} a_i(f)q^{di}. \tag{1.1.6}$$

Definition 5. *We define the* ***old subspace*** *of $S_2(\Gamma_0(N))$ to be the subspace of cusp forms that are images of degeneracy maps from all levels $M|N$. We define the* ***new subspace*** *as the space orthogonal to the old subspace under the Petersson inner product. We call a form a* ***newform*** *if it is an eigenform for all operators in the Hecke Algebra, lies in the new subspace, and if its first Fourier coefficient is normalized to one.*

The following is the main result of [5] and characterizes the structure of $S_2(\Gamma_0(N))$ in terms of new and old forms.

Theorem 2. *If f is a newform at some level N_f dividing N, then define S_f to be the subspace of $S_2(\Gamma_0(N))$ generated by all degeneracy images of f from $S_2(\Gamma_0(N_f))$ into $S_2(\Gamma_0(N))$. Then*

$$S_2(\Gamma_0(N)) = \oplus_f S_f, \tag{1.1.7}$$

where the sum is taken over all newforms f at levels $N_f|N$.

Proof. See [7] Chapter 5. □

1.1.2 Computing with Modular Forms

In this section, we give a brief description of how coefficients of newforms can be calculated. For a detailed exposition of these concepts, see Chapter 3 of [18] or Chapter 2 of [Cre97].

One can show that for all positve integers n and for all $f \in S_2(\Gamma_0(N))$,

$$a_1(T_n(f)) = a_n(f). \tag{1.1.8}$$

In the case that f is a newform, equation1.1.8 says that $a_n(f)$ is equal to the eigenvalue of T_n corresponding to f. Thus, in order to compute the Fourier coefficients of newforms, one may identify eigenspaces of the action of $\mathbf{T}$ on the new subspace of $S_2(\Gamma_0(N))$, which we will denote $S_2(\Gamma_0(N))_{\text{new}}$, and calculate the corresponding eigenvalues. The rest of this section deals with how to identify eigenspaces of a space that is isomorphic to $S_2(\Gamma_0(N))_{\text{new}}$ as a module over $\mathbf{T}$.

The Hecke algebra acts on $\mathrm{H}_1(X_0(N), \mathbb{Z})$ (see [6] pp 32 for an algebraic description or [18] 3.1 for a geometric description). For a cycle γ in $\mathrm{H}_1(X_0(N), \mathbb{Z}) \otimes \mathbb{R}$, and a form f in $S_2(\Gamma_0(N))$, the integral $2\pi i \int_\gamma f(z)dz$ defines a nondegenerate pairing

$$\langle , \rangle : S_2(\Gamma_0(N)) \times \mathrm{H}_1(X_0(N), \mathbb{Z}) \otimes \mathbb{R} \to \mathbb{C}. \tag{1.1.9}$$

This pairing is compatible with the action of $\mathbf{T}$ on $\mathrm{H}_1(X_0(N), \mathbb{Z})$ referenced above and gives a duality between $S_2(\Gamma_0(N))$ and $\mathrm{H}_1(X_0(N), \mathbb{Z}) \otimes \mathbb{R}$ as $2g$-dimensional real vector spaces, making the two spaces isomorphic as Hecke modules.

What's left is to identify the subspace of $\mathrm{H}_1(X_0(N),\mathbb{Z})\otimes\mathbb{R}$ corresponding to $S_2(\Gamma_0(N))_{\mathrm{new}}$ and compute the Hecke action on this subspace. While this may be difficult to do directly, a method called *modular symbols* can be used to compute the Hecke action on a cycle in $\mathrm{H}_1(X_0(N),\mathbb{Z})\otimes\mathbb{R}$ by considering a certain corresponding action on the initial and terminal points of the cycle's preimage in $\mathfrak{H}\cup\mathbb{Q}\cup\infty$.

Let $\mathbb{M}_2$ be the free abelian group with a basis of symbols $\{\alpha,\beta\}$ where α,β range over the elements of $\mathbb{Q}\cup\infty$, modulo the relation

$$\{\alpha,\beta\}+\{\beta,\gamma\}+\{\gamma,\alpha\}=0 \tag{1.1.10}$$

and modulo any torsion (e.g. since equation 1.1.10 implies that $3\{\alpha,\alpha\}=0$, we set $\{\alpha,\alpha\}=0$). There is an action of $\mathrm{GL}_2(\mathbb{Q})$ on $\mathbb{M}_2$: if $g\in\mathrm{GL}_2(\mathbb{Q})$, define $g(\{\alpha,\beta\})=\{g(\alpha),g(\beta)\}$. We define the *modular symbols space for* $\Gamma_0(N)$, denoted $\mathbb{M}_2(\Gamma_0(N))$, as the quotient of $\mathbb{M}_2$ by the submodule generated by elements of the form $x-g(x)$ for all $x\in\mathbb{M}_2$ and $g\in\Gamma_0(N)$, and modulo any torsion. Let $\mathbb{B}_2(\Gamma_0(N))$ be the free abelian group generated by the cusps for $\Gamma_0(N)$ and for $\beta\in\mathbb{Q}\cup\infty$, let $\{\beta\}$ denote the class of β in $\mathbb{B}_2(\Gamma_0(N))$. Let $\delta:\mathbb{M}_2(\Gamma_0(N))\to\mathbb{B}_2(\Gamma_0(N))$ be given by $\delta:\{\alpha,\beta\}\mapsto(\{\alpha\}-\{\beta\})$. Denote by $\mathbb{S}_2(\Gamma_0(N))$ the kernel of δ. Let $\phi:\mathbb{S}_2(\Gamma_0(N))\to H_1(X_0(N),\mathbb{Z})$ be the map which sends the pair $\{\alpha,\beta\}$ to the image in $X_0(N)$ of a geodecic path in path in $\mathfrak{H}$ from α to β. It was shown by Manin that the map ϕ gives an isomorphism:

$$\mathbb{S}_2(\Gamma_0(N))\cong H_1(X_0(N),\mathbb{Z}). \tag{1.1.11}$$

One can define an action of the Hecke algebra on $\mathbb{S}_2$. For p be a prime not dividing N, define

$$T_p(\{\alpha,\beta\})=\begin{pmatrix} p & 0\\ 0 & 1\end{pmatrix}\{\alpha,\beta\}+\sum_{r\ mod\ p}\begin{pmatrix} 1 & r\\ 0 & p\end{pmatrix}\{\alpha,\beta\}.$$

And for a prime $p|N$, define

$$T_p(\{\alpha,\beta\})=\sum_{r\ mod\ p}\begin{pmatrix} 1 & r\\ 0 & p\end{pmatrix}\{\alpha,\beta\}.$$

For $n>0$,

$$T_{p^{n+1}}=T_pT_{p^n}-\langle p\rangle T_{p^{n-1}}$$

If $n=\prod_i p_i^{e_i}$ gives a prime factorization of n, then we set

$$T_n=\prod_i T_{p_i^{e_i}}.$$

The Hecke action defined in this way is compatible with the isomorphism in equation 1.1.11 (as we show in the next subsection).

Theorem 3.13 and Proposition 3.21 of [18] give a finite presentation for $\mathbb{S}_2(\Gamma_0(N))$. For $d|N$, let $\phi_d:\mathbb{S}_2(\Gamma_0(N))\to\mathbb{S}_2(\Gamma_0(N))$ be given by $\phi_d(\{\alpha,\beta\})\mapsto\begin{pmatrix} d & 0\\ 0 & 1\end{pmatrix}\{\alpha,\beta\}$. One can show that the subspace of $\mathbb{S}_2(\Gamma_0(N))$ corresponding to $S_2(\Gamma_0(N))_{\mathrm{new}}$ is the intersection of the kernels of ϕ_d ranging over all $d|N$ with $d>1$. We'll call this subspace $\mathbb{S}_2(\Gamma_0(N))_{\mathrm{new}}$.

Fourier coefficients for newforms in $S_2(\Gamma_0(N))_{\mathrm{new}}$ can thus be calculated by finding a basis for $\mathbb{S}_2(\Gamma_0(N))_{\mathrm{new}}$, expressing the action of operators T_n on $\mathbb{S}_2(\Gamma_0(N))_{\mathrm{new}}$ as matrices in terms of this basis, and computing the eigenvalues of these matrices.

Decomposition of $\mathbb{S}_2(\Gamma_0(N))$ and a Hecke-Module Isomorphism with $S_2(\Gamma_0(N))$. In this section, we make explicit a correspondance between $S_2(\Gamma_0(N,\mathbb{C}))$ and a subspace of $\mathbb{S}_2(\Gamma_0(N))\otimes\mathbb{C}$. To this end, we present a useful decomposition of $\mathbb{S}_2(\Gamma_0(N))\otimes\mathbb{C}$.

Following the notation in [13], let $\overline{S_2(\Gamma_0(N,\mathbb{C}))}$ denote the space of antiholomorphic modular forms and consider the map $i : f(z)\mapsto f(-\bar{z})$. Denote by η the element $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ in $M_2(\mathbb{Q})$. Since η normalizes $\Gamma_0(N)$, then for any cusp form f and any $\gamma = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ in $\Gamma_0(N)$, we have $\underline{i(f(\gamma(z)))} = i(f(\eta^{-1}\gamma\eta(z))) = (c\bar{z}+d)^2 f(z)$, so the map i is an isomorphism from $S_2(\Gamma_0(N,\mathbb{C}))$ to $\overline{S_2(\Gamma_0(N,\mathbb{C}))}$. We define the so-called *star involution* on $\mathbb{S}_2(\Gamma_0(N))\otimes\mathbb{C}$ as $i^* : \{\alpha,\beta\}\mapsto\{-\alpha,-\beta\}$.

Let $\mathbb{S}_2(\Gamma_0(N))^+$ denote the subspace of $\mathbb{S}_2(\Gamma_0(N))$ invariant under the action of i^* (e.g. consisting of symbols $\{\alpha,\beta\}$ such that $\{\alpha,\beta\}$ and $i^*(\{\alpha,\beta\}) = \{-\alpha,-\beta\}$ are $\Gamma_0(N)$-equivalent), and let $\mathbb{S}_2(\Gamma_0(N))^-$ be the subspace anti-invariant under i^*. We will make use of the decomposition $\mathbb{S}_2(\Gamma_0(N)) = \mathbb{S}_2(\Gamma_0(N))^+\oplus\mathbb{S}_2(\Gamma_0(N))^-$.

Consider the pairing $\langle,\rangle : (S_2(\Gamma_0(N,\mathbb{C}))\oplus\overline{S_2(\Gamma_0(N,\mathbb{C}))})\times\mathbb{S}_2\Gamma_0(N))\to\mathbb{C}$ given by $\langle f+g,\{\alpha,\beta\} = \int_\alpha^\beta f(z)dz+\int_\alpha^\beta g(z)d\bar{z}$. Theorem 3 in [13] shows that this pairing is nondegenerate.

Since

$$\begin{aligned}
\langle i(f+g),\{\alpha,\beta\}\rangle &= -\int_\alpha^\beta f(-\bar{z})d\bar{z}+\int_\alpha^\beta g(-\bar{z})dz && (1.1.12)\\
&= \int_{-\alpha}^{-\beta} f(z)dz+\int_{-\alpha}^{-\beta} g(z)d\bar{z} && (1.1.13)\\
&= \langle f+g, i^*(\{\alpha,\beta\})\rangle, && (1.1.14)\\
& && (1.1.15)
\end{aligned}$$

the map i^* is adjoint to i with respect to $\langle,\rangle$.

This discussion leads to:

Proposition 1. *$\langle,\rangle$ induces a nondegenerate pairing $S_2(\Gamma_0(N,\mathbb{C}))\times\mathbb{S}_2(\Gamma_0(N))^+\to\mathbb{C}$.*

Proof. Define $\langle,\rangle' : S_2(\Gamma_0(N,\mathbb{C}))\times\mathbb{S}_2(\Gamma_0(N))^+\to\mathbb{C}$ by

$$\begin{aligned}
\langle f,\{\alpha,\beta\}\rangle' &= \langle f+i(f),\{\alpha,\beta\}\rangle && (1.1.16)\\
&= \langle f,\{\alpha,\beta\}\rangle+\langle f, i^*(\{\alpha,\beta\}\rangle && (1.1.17)\\
&= 2\langle f,\{\alpha,\beta\}\rangle && (1.1.18)\\
& && (1.1.19)
\end{aligned}$$

□

Theorem 1.42 in [18] shows that the Hecke operators T_n, as defined on the respective spaces, are self-adjoint with respect to the pairings $\langle,\rangle$ and $\langle,\rangle'$. Thus we have a Hecke module isomorphism ϕ from $\mathbb{S}_2(\Gamma_0(N))^+\otimes\mathbb{C}$ to the dual of $S_2(\Gamma_0(N,\mathbb{C}))$ (denoted $S_2(\Gamma_0(N,\mathbb{C}))^*$) given by $\phi : s\mapsto\langle\cdot,s\rangle'$ for $s\in\mathbb{S}_2(\Gamma_0(N)^+$.

We use the map ϕ as defined above to establish a correspondance between $S_2(\Gamma_0(N,\mathbb{C}))$ and $\mathbb{S}_2(\Gamma_0(N))^+\otimes\mathbb{C}$ as Hecke modules as follows. Decompose $S_2(\Gamma_0(N,\mathbb{C}))$ as a direct sum of $\mathbf{T}\otimes\mathbb{C}$-invariant submodules: $S_2(\Gamma_0(N,\mathbb{C})) = \oplus_i S_i$. Then define $S_j^*\subset S_2(\Gamma_0(N,\mathbb{C}))^*$ as $Ann_{S_2(\Gamma_0(N,\mathbb{C}))^*}(\oplus_{i\neq j}S_i)$.

Proposition 2. *For any j, S_j is isomorphic to S_j^* as Hecke modules.*

Proof. The two spaces are isomorphic $\mathbb{C}$-vector spaces since they are of the same dimension; the isomorphism as Hecke modules follows from the natural action of $\mathbf{T} \otimes \mathbb{C}$ on the functionals $\{\gamma : f \mapsto c | f \in S_2(\Gamma_0(N, \mathbb{C})), c \in \mathbb{C}\}$ given by $t(\gamma(f)) = \gamma(t(f))$. $\square$

The isomorphism above is non-cannonical, thus it does not indicate a computable Hecke-compatible map between forms in $S_2(\Gamma_0(N, \mathbb{C}))$ and symbols in $\mathbb{S}_2(\Gamma_0(N))^+ \otimes \mathbb{C}$. It does however show how to associate to each $\mathbf{T} \otimes \mathbb{C}$-invariant subspace $S_i \subset S_2(\Gamma_0(N, \mathbb{C}))$ a corresponding subspace in $\mathbb{S}_2(\Gamma_0(N))^+ \otimes \mathbb{C}$, namely: $\phi^{-1}(S_i^*)$.

1.1.3 Modular Elliptic Curves

The modular curve $X_0(N)$ has a canonical model over $\mathbb{Q}$, as we describe now. Consider the elliptic curve $E/\mathbb{Q}(j) = y^2 + xy = x^3 - \frac{36}{j-1728}x - \frac{1}{j-1728}$. Let C be any subgroup of $E(\overline{\mathbb{Q}(j)})$ of order N. Let F_N be the smallest extension of $\mathbb{Q}(j)$ such that $\sigma(C) = C$ for all $\sigma \in \mathrm{Gal}(\overline{\mathbb{Q}(j)}/F_N)$. Then the field F_N is isomorphic to the function field of a curve $X/\mathbb{Q}$. The solution X to this moduli problem is canonical in the sense that it is not dependent on the subgroup C. The curve X exhibits the propety that $X(\mathbb{Q})$ is isomorphic to the set of rational points of $X_0(N)$ and is taken to be the model of $X_0(N)$ over $\mathbb{Q}$. We denote X by $X_0N)$ again for ease of notation.

The group $\mathrm{H}_1(X_0(N), \mathbb{Z})$ injects into $\mathrm{Hom}(S_2(\Gamma_0(N)), \mathbb{C})$ via the identification $\gamma \mapsto (f \mapsto \int_\gamma 2\pi f\, dz)$ for $\gamma \in \mathrm{H}_1(X_0(N), \mathbb{Z})$ and $f \in S_2(\Gamma_0(N))$.

Denote by $J_0(N)$ the Jacobian variety of $X_0(N)$ over $\mathbb{Q}$. Thus $J_0(N)$ is an abelian variety over $\mathbb{Q}$ whose group of complex points is $\mathrm{Pic}^0(X_0(N))$. The isomorphism given by the Abel-Jacobi map

$$\mathrm{Pic}^0(X_0(N), \mathbb{C}) \cong \mathrm{Hom}(S_2(\Gamma_0(N)), \mathbb{C})/\mathrm{H}_1(X_0(N), \mathbb{Z})$$

defines an action of $\mathbf{T}$ on $J_0(N)$.

Let f be a newform of level N with integral coefficients. Let $I_f = Ann_{\mathbf{T}}(f)$. We associate to f an abelian variety A_f over $\mathbb{Q}$ defined by

$$A_f = J_0(N)/I_f J_0(N). \tag{1.1.20}$$

If f has integral coefficients, then A_f is an elliptic curve over $\mathbb{Q}$ (see [6]).

Definition 6. *If f has integral Fourier coefficients, an elliptic curve E constructed as A_f is called a* ***modular elliptic curve****. We also say that E is* ***the curve associated to f*** *or that f is* ***the newform associated to E***

The following theorem was once known as the Taniyma-Shimura conjecture. It is now called the Modularity Theorem. See the book [7] for many similar statements from geometric, algebraic, topological, and complex-analytic perspectives.

Theorem 3 (Breuil, Conrad, Diamond, Taylor)**.** *An elliptic curve over the rational field is isogenous to an elliptic curve of the form A_f for some newform f with integer coefficients.*

1.1.4 Congruence Primes

Definition 7. *Let p be a prime and let X and Y be two subspaces of $S_2(\Gamma_0(N))$. We call p a* ***congruence prime linking** X **and** Y if there exist forms $f \in X$ and $g \in Y$ with integer Fourier coefficients which are not both multiples of p such that $a_n(f) \equiv a_n(g)$ mod p for all n. In this case, we write $f \equiv g$ mod p. If f is a cusp form, then we also say that p is a congruence prime linking f and Y in the event that p is a congruence prime linking the span of f to Y.*

Perhaps surprisingly, congruence primes linking spaces X and Y are quite common. In fact, it happens that there always exist primes of congruence linking any two complementary subspaces of $S_2(\Gamma_0(N))$. (This follows from [12] proposition 10.6.)

The article [14] gives several characterizations about congruence primes linking complementary subspaces of $S_2(\Gamma_0(N))$. We are often interested in congruence primes linking a single newform to the rest of the space $S_2(\Gamma_0(N))$. The article [10] gives several characterizations of such primes.

Lemma 1. *Let f be a newform with integer coefficients. The prime p is a congruence prime linking the subspace generated by f to a subspace Y in $S_2(\Gamma_0(N))$ if and only if f is congruent modulo p to a cusp form with integer Fourier coefficients in Y.*

Proof. In the “if" direction, the result follows by definition. Conversely, if p is a congruence prime linking the $\mathbb{C}$-span of f to the subspace Y, then there exists some cf and $g \in Y$ with integer coefficients such that $cf \equiv g$ mod p. We can assume that c is a unit modulo p because if $p|c$, then cf and g would both be multiples of p. Let $d \in \mathbb{Z}$ be such that $dc \equiv 1$ mod p. Then $f \equiv dg$ mod p. $\square$

1.2 The Birch and Swinnerton-Dyer Conjecture

In this section, we state the conjectural Birch and Swinnerton-Dyer formula and describe the elliptic curve invariants that occur therein: the L-function, the Tate-Shaferavich group, the elliptic regulator, the tamagawa product, and the real period.

Let K be a local field that is complete with respect to a valuation ν. Let E/K be a curve given by

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{1.2.1}$$

We say that equation 1.2.1 is a *minimal Weierstrass equation* for E at ν if

1. $\nu(a_i) \geq 0$ for all i

2. The discriminant of equation 1.2.1 is minimal under the restriction in item 1.

Let F is a number field and let ν be any finite or archimedean prime. Then we say that an equation for E/F is minimal at ν if the same equation is minimal over F_ν. In the case that an equation for E/F is minimal at all places ν, we call the equation a *global minimal Weierstrass equation* for E and denote the equation $E_{\min}$. It can be shown that all elliptic curves over $\mathbb{Q}$ have a global minimal Weierstrass equation.

Let $E/\mathbb{Q}$ be an elliptic curve defined over the rationals. The curve E is said to have *good reduction* at p if the reduction of a global minimal Weierstrass equation for E modulo p, denoted $\tilde{E}$, has no singular points. We say E has *multiplicative reduction* at p if $\tilde{E}$ has a node. If E is of multiplicative reduction and the slopes of the tangent lines at the node in $\tilde{E}$ are both elements of

$\mathbb{F}_p$ (respectively, are *not* both elements of $\mathbb{F}_p$), we say that E is of *split reduction* (respectively, we say that E is of *non-split reduction*). Finally, if $\tilde{E}$ has a cusp, we say E has *additive reduction* at p. These definitions are independant of the global minimal Weierstrass equation chosen for E. In the case that E has good reduction at p, define $a_p = p+1-|\tilde{E}(\mathbb{F}_p)|$.

Definition 8. *Let p be a prime. We define the* ***local L-factor of E at p****, denoted $L(E/\mathbb{Q}_p,s)$, to be*

- $(1-a_p p^{-s}+p^{1-2s})^{-1}$ *if E has good reduction at p,*
- $(1-p^{-s})^{-1}$ *if E has split reduction at p,*
- $(1+p^{-s})^{-1}$ *if E has non-split reduction at p,*
- 1 *if E has additive reduction at p.*

Definition 9. *We define the* ***global L-function of E****, denoted $L(E/\mathbb{Q},s)$, by*

$$L(E/\mathbb{Q},s) = \prod_p L(E/\mathbb{Q}_p,s) \tag{1.2.2}$$

where p ranges over the prime numbers.

Let $E/\mathbb{Q}$ be an elliptic curve. If r is a rational number, let $M(r)$ denote $max(|s|,|t|)$ where s and t are integers such that r=s/t and (s,t)=1. If $f: E(\mathbb{Q}) \to \mathbb{Q}$ is an even function, then let h_f be a function on $E(\mathbb{Q})$ given by

$$h_f(P) = \log(M(f(P)) \tag{1.2.3}$$

for $P \neq 0$.

Let $f: E(\mathbb{Q}) \to \mathbb{Q}$ be an even function. Define $\hat{h}: E(\mathbb{Q}) \mapsto \mathbb{R}$ by

$$\hat{h}(P) = \frac{1}{deg(f)} \lim_{N\to\infty} \frac{h_f(2^N P)}{4^N}. \tag{1.2.4}$$

The function $\hat{h}$ does not depend on the function f (see [17] lemma 6.3). The *Neron-Tate pairing* $\langle \cdot,\cdot \rangle$ is given by

$$\langle P,Q \rangle = \hat{h}(P+Q) - \hat{h}(P) - \hat{h}(Q) \tag{1.2.5}$$

Definition 10. *Let $\{P_1 ... P_r\}$ be a set of generators for $E(\mathbb{Q})/E_{\text{tors}}(\mathbb{Q})$. The* ***elliptic regulator****, $\mathrm{R}(E/\mathbb{Q})$, is the determinant of the matrix whose ij-th entry is $\langle P_i, P_j \rangle$.*

The *invariant differential* of a Weierstrass equation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ is $dx/(2y + a_1x + a_3)$.

Let E be an elliptic curve over $\mathbb{Q}$. Let ω be the invariant differential of a global minimal Wierstrass equation $E_{\min}$ for E (i.e. with minimal discriminant everywhere and integral coefficients). The *real period* of E, denoted Ω, is the quantity $\int_{E_{\min}(R)} \omega$.

For a prime p, let $E_0(\mathbb{Q}_p)$ denote the set of points $P \in E(\mathbb{Q}_p)$ which map to nonsingular points of $\tilde{E}(\mathbb{F}_p)$ when reduced modulo p. The group $E_0(\mathbb{Q}_p)$ is of finite index in $E(\mathbb{Q}_p)$(see [17] chapter VIII).

Definition 11. *If p is a prime number, then the* ***Tamagawa number at p****, denoted c_p, is the order* $|E(\mathbb{Q}_p)/E_0(\mathbb{Q}_p)|$

It follows immediately from definition that $c_p = 1$ for all primes p of good reduction.

Definition 12. *We define the* ***Tate-Shaferavich group*** $\text{Ш}(E/\mathbb{Q})$ *as the group*

$$\text{Ш}(E/\mathbb{Q}) = \ker\left\{\mathrm{H}^1(\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q}), E(\overline{\mathbb{Q}})) \to \prod_p \mathrm{H}^1(\mathrm{Gal}(\overline{\mathbb{Q}_p}/\mathbb{Q}_p), E(\overline{\mathbb{Q}_p}))\right\}. \tag{1.2.6}$$

For more information and some geometric interpretations of $\text{Ш}(E/\mathbb{Q})$, see [17] Chapter X.

Let $E/\mathbb{Q}$ be an elliptic curve. Let $L^r(E/\mathbb{Q},1)$ denote the r-th derivative of $L(E/\mathbb{Q},s)$ evaluated at $s = 1$.

Conjecture 1 (Birch and Swinnerton-Dyer)**.**

$$\frac{L^r(E/\mathbb{Q},1)}{r!\,\Omega} = \frac{|\text{Ш}(E/\mathbb{Q})| \cdot \mathrm{R}(E/\mathbb{Q}) \cdot \prod_p c_p}{|E_{\text{tors}}(\mathbb{Q})|^2} \tag{1.2.7}$$

We call equation 1.2.7 the *conjectural Birch and Swinnerton-Dyer formula*.

1.3 Motivation for this Thesis: Cancellations in the Conjectural Birch and Swinnerton-Dyer Formula

In this section, let $E/\mathbb{Q}$ be a modular elliptic curve associated to the newform f of level N. The right hand side of equation 1.2.7 has been studied extensively (see the introduction in [11]). In particular, in regard to the relationship between the Tamagawa product $\Pi_p c_p$ and the order of $E_{\text{tors}}(\mathbb{Q})$, M. Emerton [8] showed that when N is prime, $\Pi_p c_p = |E_{\text{tors}}(\mathbb{Q})|$.

When N is not prime, $|E_{\text{tors}}(\mathbb{Q})|$ need not equal $\Pi_p c_p$. In fact, this first occurs when $N = 42$. D. Lorenzini [11] has shown that if $\ell > 3$ is a prime number, then the order of the ℓ-primary part of $E_{\text{tors}}(\mathbb{Q})$ divides $\Pi_p c_p$.

In the other direction, Agashe [1] has conjectured that:

Conjecture 2 (Agashe)**.** *If an odd prime ℓ divides $\Pi_p c_p$, then either ℓ divides the order of $E_{\text{tors}}(\mathbb{Q})$ or the newform f is congruent modulo ℓ to a form in the old space.*

The following partial result towards conjecture 2 is given in [1].

Proposition 3. *Let ℓ be an odd prime such that either $\ell \nmid N$ or for all primes r that divide N, $\ell \nmid (r-1)$. If ℓ divides the order of the geometric component group of E at p for some prime $p||N$, then either $E[\ell]$ is reducible or the newform f is congruent to a newform of level dividig N/p (for all Fourier coefficients whose indices are coprime to $N\ell$) modulo a prime ideal over ℓ in a number field containing the Fourier coefficients of both newforms.*

In fact, in [1] the author conjectures more specifically that:

Conjecture 3 (Agashe)**.** *If an odd prime ℓ divides c_p for some prime p, then either ℓ divides the order of $E_{\text{tors}}(\mathbb{Q})$ or the newform f is congruent modulo ℓ to a form in the subspace generated by degeneracy map images of newforms of levels dividing N/p.*

Agashe was able to test these conjectures using the SAGE [19] command congruence_number() for curves of conductor up to 1000, at which point computations became too slow to gather more data. Since many interesting cases did not lie in this range, more data was desired and it was decided that a more efficient algorithm for computing congruence numbers would be useful for this and other purposes. The question of how to develop such an algorithm was the motivation for this thesis.

The first idea was to compute intersection numbers between subspaces of $S_2(\Gamma_0(N))$ by calculating orders of certain quotients of $\mathrm{H}_1(X_0(N), \mathbb{Z})$. Chapter two of this thesis is about this method and its implementation. The performance of this approach was not as fast as was hoped for, so we tried other ideas (see Chapter three) until we found a method that performed significantly better (see Table 3.3 in Section 3.2 for a comparison). The method we use involves computing q-expansions for modular forms to high precision and may have applications elsewhere.

CHAPTER 2

INTERSECTION NUMBERS BETWEEN SPACES OF CUSPIDAL MODULAR FORMS

In this chapter, we define intersection numbers and prove a result relating intersection numbers between spaces of cuspidal modular forms to the cooresponding annihilator ideals in the Hecke algebra $\mathbf{T} \otimes \mathbb{Q}$. We describe an algorithm for computing intersection numbers using this result.

2.1 Introduction

Throughout this chapter, take $N > 5$ to be an integer. Take g to be a newform of level N_g dividing N and let $[g]$ be its Galois conjugacy class. Denote by $S_{([g],\mathbb{C})}$ the subspace in $S_2(\Gamma_0(N),\mathbb{C})$ generated by degeneracy map images of forms in $[g]$. Denote by $S_{([g],\mathbb{Q})}$ the $\mathbb{Q}$ vector space of all such forms with rational Fourier coefficients. Then $S_2(\Gamma_0(N),\mathbb{Q}) = \oplus_{[g]} S_{([g],\mathbb{Q})}$ where the sum is over the Galois conjugacy classes of newforms at levels dividing N.

If $X = \oplus_{[g]} S_{([g],\mathbb{C})}$ and $Y = \oplus_{[f]} S_{([f],\mathbb{C})}$ are two subpaces of $S_2(\Gamma_0(N),\mathbb{C})$, we define the *intersection number between X and Y* as the order of intersection between the two associated abelian varieties $(J_0(N)/\mathrm{Ann}_{\mathbf{T}}(X))^{\vee}$ and $(J_0(N)/\mathrm{Ann}_{\mathbf{T}}(Y))^{\vee}$. One reason for studying intersection numbers between such spaces X and Y as a means to study congruence primes between X and Y, since [4] shows that the congruence primes also divide the intersection number.

Given a basis $B = \{s_1 \dots s_{2g}\}$ for $\mathrm{H}_1(X_0(N),\mathbb{Z})$, the *matrix for to the Hecke operator t relative to the basis B* is the unique matrix M that satisfies

$$t(s_i) = \sum_{j=1}^{2g} M_{ij}(s_j). \tag{2.1.1}$$

Alternatively, taking $\mathrm{H}_1(X_0(N),\mathbb{Z})$ as a $\mathbb{Z}[\mathbf{T}]$ module, we get a linear representation $\rho_L : \mathbf{T} \to \mathrm{GL}_{2g}(\mathbb{Z})$. Choosing a basis B for $\mathrm{H}_1(X_0(N),\mathbb{Z})$ induces an associated matrix representation, ρ. In this context, the matrix for the Hecke operator t is $\rho(t)$.

If M is a matrix with rational entries, then let d denote the least common multiple of all of the denominators of entries of M. We denote by M_{int} the matrix given by $d \cdot M$, the integer matrix produced by clearing denominators in M.

If M is an $n \times m$ matrix with integer coefficients, we define the *torsion order* of M to be the product of the diagonal entries of the Smith Normal Form of M. We denote the torsion order of M

as $\operatorname{tors}(M)$. The torsion order of M is so called because it is the order of the torsion part of the group $\mathbb{Z}^m/\operatorname{Col}(M)$ where $\operatorname{Col}(M)$ the denotes subgroup of $\mathbb{Z}^m$ generated by the columns of M.

Given a set of $n \times m$ matrices, $\{M_1, \ldots, M_k\}$, we denote by $M_1 \ldots M_k$ the $n \times (mk)$ matrix produced by the horizontal augmentation of the $\{M_i\}$.

We have the following result.

Theorem 4. *Let $X = \oplus_{[g]} S_{([g],\mathbb{C})}$ and $Y = \oplus_{[f]} S_{([f],\mathbb{C})}$ be two disjoint and complementary subspaces of $S_2(\Gamma_0(N))$. Let X_H and Y_H denote the images of $\oplus_{[g]} S_{([g],\mathbb{Q})}$ and $\oplus_{[f]} S_{([f],\mathbb{Q})}$ respectively in $\mathrm{H}_1(X_0(N),\mathbb{Z}) \otimes \mathbb{Q}$ under the isomorphism described in section 1.1.2. Let $\{t_1, \ldots, t_m\}$ and $\{r_1, \ldots, r_m\}$ be generators for the annihilator ideals in $\mathbf{T}_{\mathbb{Z}} \otimes \mathbb{Q}$ of X_H, Y_H respectively . Let $\{T_1 \ldots T_m\}$ and $\{R_1 \ldots R_n\}$ be matrices for $\{t_1, \ldots, t_m\}$ and $\{r_1, \ldots r_n\}$ respectively relative to some basis for $\mathrm{H}_1(X_0(N),\mathbb{Z})$. Then the intersection number between X and Y is*

$$\frac{\operatorname{tors}(T_{1_{\mathrm{int}}} \ldots T_{m_{\mathrm{int}}} R_{1_{\mathrm{int}}} \ldots R_{n_{\mathrm{int}}})}{\operatorname{tors}(T_{1_{\mathrm{int}}} \ldots T_{m_{\mathrm{int}}}) \cdot \operatorname{tors}(R_{1_{\mathrm{int}}} \ldots R_{n_{\mathrm{int}}})} \tag{2.1.2}$$

In section 2.2, we prove Theorem 1. In section 3.2, we present an algorithm for computing intersection numbers, and this section can be read independantly of section 2.2.

2.2 Proof of Theorem 1

If X is a subspace of $S_2(\Gamma_0(N))$ of the form $\oplus_{[g]} S_{([g],\mathbb{C})}$, then denote by I_X the annihilator ideal of $\oplus_{[g]} S_{([g],\mathbb{Q})}$ in the Hecke algebra $\mathbf{T}_{\mathbb{Z}}$. Denote by $I_{(X,\mathbb{Q})}$ the annihilator ideal of $\oplus_{[g]} S_{([g],\mathbb{Q})}$ in $\mathbf{T}_{\mathbb{Z}} \otimes \mathbb{Q}$.

Lemma 2. *If X and Y are disjoint subspaces of $S_2(\Gamma_0(N))$ both of the form $\oplus_{[g]} S_{([g],\mathbb{C})}$, then the intersection number between X and Y is the order of the group*

$$\frac{\mathrm{H}_1(X_0(N),\mathbb{Z})}{\mathrm{H}_1(X_0(N),\mathbb{Z})[I_X] + \mathrm{H}_1(X_0(N),\mathbb{Z})[I_Y]} \tag{2.2.1}$$

Proof. This is Lemma 4.1 of [2]. □

Lemma 1 is the basis for our computation. However, directly finding generators for I_X and I_Y is computationally difficult. The remaining lemmas in the proof of Theorem 1 show how to modify Lemma 1 so that one may instead use generators for $I_{(X,\mathbb{Q})}$ and $I_{(Y,\mathbb{Q})}$, which are less costly to compute.

For later reference, we now establish a small fact regarding group orders.

Lemma 3. *Let G be a finitely generated abelian group, and let H_1 and H_2 be subgroups of G. If $K_1 \subseteq H_1$ and $K_2 \subseteq H_2$ are subgroups of finite index in H_1 and H_2 respectively, then*

$$\left|\frac{H_1 + H_2}{K_1 + K_2}\right| = \frac{|H_1/K_1| \cdot |H_2/K_2|}{|(H_1 \cap H_2)/(K_1 \cap K_2)|} \tag{2.2.2}$$

Proof. Consider the following commutative diagram of exact sequences.

$$\begin{tikzcd}
 & 0 \arrow[d] & 0 \arrow[d] & 0 \arrow[d] & \\
0 \arrow[r] & K_1 \cap K_2 \arrow[r] \arrow[d] & K_1 \times K_2 \arrow[r] \arrow[d] & K_1 + K_2 \arrow[r] \arrow[d] & 0 \\
0 \arrow[r] & H_1 \cap H_2 \arrow[r] \arrow[d] & H_1 \times H_2 \arrow[r] \arrow[d] & H_1 + H_2 \arrow[r] \arrow[d] & 0 \\
 & \frac{H_1 \cap H_2}{K_1 \cap K_2} \arrow[d] & \frac{H_1 \times H_2}{K_1 \times K_2} \arrow[d] & \frac{H_1 + H_2}{K_1 + K_2} \arrow[d] & \\
 & 0 & 0 & 0 &
\end{tikzcd}$$

The Nine Lemma completes the bottom row so that

$$0 \longrightarrow \frac{H_1 \cap H_2}{K_1 \cap K_2} \longrightarrow \frac{H_1 \times H_2}{K_1 \times K_2} \longrightarrow \frac{H_1 + H_2}{K_1 + K_2} \longrightarrow 0 \tag{2.2.3}$$

is exact. In particular

$$\frac{H_1 + H_2}{K_1 + K_2} \cong \frac{(H_1 \times H_2)/(K_1 \times K_2)}{(H_1 \cap H_2)/(K_1 \cap K_2)}. \tag{2.2.4}$$

The result follows. □

The following lemma will be used to help calculate the order of the group in equation (2.2.1). For a group G, denote the order of the torsion subgroup of G as $\mathrm{tors}(G)$.

Lemma 4. *Let G be a finitely generated abelian group, and let H_1 and H_2 be subgroups of G such that G/H_1 and G/H_2 are both torsion-free and $\frac{G}{H_1+H_2}$ is finite. If $K_1 \subseteq H_1$ and $K_2 \subseteq H_2$ are subgroups of finite index in H_1 and H_2 respectively, then*

$$\left|\frac{G}{H_1 + H_2}\right| = \left|\frac{G}{K_1 + K_2}\right| \cdot \frac{\mathrm{tors}(G/(K_1 \cap K_2))}{\mathrm{tors}(G/K_1) \cdot \mathrm{tors}(G/K_2)} \tag{2.2.5}$$

Proof. By the Third Isomorphism Theorem we have

$$\left|\frac{G}{H_1 + H_2}\right| = \frac{\left|\frac{G}{K_1+K_2}\right|}{\left|\frac{H_1+H_2}{K_1+K_2}\right|} \tag{2.2.6}$$

However, by Lemma 3 above,

$$\left|\frac{H_1 + H_2}{K_1 + K_2}\right| = \frac{|H_1/K_1| \cdot |H_2/K_2|}{|(H_1 \cap H_2)/(K_1 \cap K_2)|} \tag{2.2.7}$$

so that equation 2.2.6 becomes

$$\left|\frac{G}{H_1 + H_2}\right| = \frac{|G/(K_1 + K_2)| \cdot |(H_1 \cap H_2)/(K_1 \cap K_2)|}{|H_1/K_1| \cdot |H_2/K_2|}. \tag{2.2.8}$$

By the Third Isomorphism Theorem, for both $i = 1$ and $i = 2$ we have

$$G/H_i \cong \frac{G/K_i}{H_i/K_i} \tag{2.2.9}$$

as well as

$$G/(H_1 \cap H_2) \cong \frac{G/(K_1 \cap K_2)}{(H_1 \cap H_2)/(K_1 \cap K_2)} \tag{2.2.10}$$

Since, G/H_i are torsion-free, equation 2.2.9 says that $|H_i/K_i| = \mathrm{tors}(G/K_i)$. Likewise, equation 2.2.10 indicates $|(H_1 \cap H_2)/(K_1 \cap K_2)| = \mathrm{tors}(G/(K_1 \cap K_2)$. So equation 2.2.8 becomes

$$\left|\frac{G}{H_1 + H_2}\right| = \frac{|G/(K_1+K_2)| \cdot \mathrm{tors}(G/(K_1 \cap K_2))}{\mathrm{tors}(G/K_1) \cdot \mathrm{tors}(G/K_2)} \tag{2.2.11}$$

as desired. □

Similar to the notation for matrices defined in the introduction, if I is an ideal in $\mathbf{T}_{\mathbb{Z}} \otimes \mathbb{Q}$ with generators $\{t_1, \ldots, t_n\}$, we will write I_{int} to denote the ideal in $\mathbf{T}_{\mathbb{Z}}$ generated by $\{z_1 t_1, \ldots, z_n t_n\}$, where $z_i \in \mathbb{Z}$ is the smallest positive integer such that $z_i t_i \in \mathbf{T}_{\mathbb{Z}}$. In this case, we will also denote $z_i t_i$ by $t_{i_{int}}$. (Note that I_{int} depends on the choice of generators $\{t_i\}$ and on the integers $\{z_i\}$, so there does not exist a one-to-one correspondance between ideals in $\mathbf{T} \otimes \mathbb{Q}$ and ideals in $\mathbf{T}_{\mathbb{Z}}$ of the form I_{int}. However, we supress this fact in the following because this dependance does effect the results as stated.)

Lemma 5. *If X and Y are disjoint, complementary subspaces of $S_2(\Gamma_0(N))$ both of the form $\oplus_{[g]} S_{([g],\mathbb{C})}$, Then $I_{(X,\mathbb{Q})_{\mathrm{int}}} \mathrm{H}_1(X_0(N),\mathbb{Z})$ is a subgroup of finite index in $\mathrm{H}_1(X_0(N),\mathbb{Z})[I_Y]$ and $I_{(Y,\mathbb{Q})_{\mathrm{int}}} \mathrm{H}_1(X_0(N),\mathbb{Z})$ is a subgroup of finite index in $\mathrm{H}_1(X_0(N),\mathbb{Z})[I_X]$. And the order of the group in equation (2.2.1) is equal to*

$$\frac{\left|\frac{\mathrm{H}_1(X_0(N),\mathbb{Z})}{I_{(Y,\mathbb{Q})_{\mathrm{int}}}\mathrm{H}_1(X_0(N),\mathbb{Z}) + I_{(X,\mathbb{Q})_{\mathrm{int}}}\mathrm{H}_1(X_0(N),\mathbb{Z})}\right|}{\mathrm{tors}\left(\frac{\mathrm{H}_1(X_0(N),\mathbb{Z})}{I_{(X,\mathbb{Q})_{\mathrm{int}}}\mathrm{H}_1(X_0(N),\mathbb{Z})}\right) \cdot \mathrm{tors}\left(\frac{\mathrm{H}_1(X_0(N),\mathbb{Z})}{I_{(Y,\mathbb{Q})_{\mathrm{int}}}\mathrm{H}_1(X_0(N),\mathbb{Z})}\right)} \tag{2.2.12}$$

Proof. It is obvious that $I_{(X,\mathbb{Q})}(\mathrm{H}_1(X_0(N),\mathbb{Z}) \otimes \mathbb{Q}$ is isomorphic to $(\mathrm{H}_1(X_0(N),\mathbb{Z}) \otimes \mathbb{Q})[I_{(Y,\mathbb{Q})}]$, since they are both just Y_H. It is not difficult to see that this induces an isomorphism between $I_X \mathrm{H}_1(X_0(N),\mathbb{Z}) \otimes \mathbb{Q}$ and $\mathrm{H}_1(X_0(N),\mathbb{Z})[I_Y] \otimes \mathbb{Q}$. Since the two spaces are equal upon tensoring with $\mathbb{Q}$, and $I_X \mathrm{H}_1(X_0(N),\mathbb{Z}) \subseteq \mathrm{H}_1(X_0(N),\mathbb{Z})[I_Y]$, $I_X \mathrm{H}_1(X_0(N),\mathbb{Z})$ is a subgroup of finite index in $\mathrm{H}_1(X_0(N),\mathbb{Z})[I_Y]$. Likewise, $I_Y \mathrm{H}_1(X_0(N),\mathbb{Z})$ is a subgroup of finite index in $\mathrm{H}_1(X_0(N),\mathbb{Z})[I_X]$.

□

We will now show how the quanties in the numerator and denominator of equation (2.2.12) are equal to those occuring in equation (2.1.2), which will prove Theorem 1. Let $B = \{s_1, \ldots s_{2g}\}$ be a basis for $\mathrm{H}_1(X_0(N),\mathbb{Z})$. Let $\{t_1, \ldots, t_m\}$ and $\{r_1, \ldots, r_n\}$ be generators $I_{(X,\mathbb{Q})}$ and $I_{(Y,\mathbb{Q})}$ respectively. Let $\{T_1 \ldots T_m\}$ and $\{R_1 \ldots R_n\}$ be matrices for $\{t_1, \ldots, t_n\}$ and $\{r_1, \ldots r_m\}$ relative to B.
The group $\frac{\mathrm{H}_1(X_0(N),\mathbb{Z})}{I_{(X,\mathbb{Q})_{\mathrm{int}}}\mathrm{H}_1(X_0(N),\mathbb{Z})}$ has a presentation

$$\langle s_1 \ldots s_{2g} | t_{i_{int}}(s_j) \text{ for } i = 1 \ldots m, j = 1 \ldots 2g \rangle. \tag{2.2.13}$$

The group $\frac{H_1(X_0(N),\mathbb{Z})}{I_{(Y,\mathbb{Q})_{\text{int}}} H_1(X_0(N),\mathbb{Z})}$ has a presentation

$$\langle s_1 \dots s_{2g} | r_{i_{int}}(s_j) \text{ for } i = 1 \dots n, j = 1 \dots 2g \rangle. \tag{2.2.14}$$

The group $\frac{H_1(X_0(N),\mathbb{Z})}{I_{(Y,\mathbb{Q})_{\text{int}}} H_1(X_0(N),\mathbb{Z}) + I_{(X,\mathbb{Q})_{\text{int}}} H_1(X_0(N),\mathbb{Z})}$ has a presentation

$$\langle s_1 \dots s_{2g} | t_{i_{int}}(s_j) \text{ for } i = 1 \dots m, j = 1 \dots 2g, r_{i_{int}}(s_j) \text{ for } i = 1 \dots n, j = 1 \dots 2g \rangle. \tag{2.2.15}$$

Let e_i denote the i-th standard basis vector (with n entries, a 1 in the i-th position and 0's elswhere). Then using the map $s_i \mapsto e_i$, the group presented in equation (2.2.13) is isomorphic to $\mathbb{Z}^n/\text{Col}(T_{1_{\text{int}}} \dots T_{m_{\text{int}}})$, the order of which is precisely $\text{tors}(T_{1_{\text{int}}} \dots T_{m_{\text{int}}})$. Similarly, the groups in (2.2.14) and (2.2.15) can be computed as $\text{tors}(T_{1_{\text{int}}} \dots T_{m_{\text{int}}} R_{1_{\text{int}}} \dots R_{n_{\text{int}}})$ and $\text{tors}(R_{1_{\text{int}}} \dots R_{n_{\text{int}}})$ respectively. Theorem 4 follows.

2.3 The Algorithm

Theorem 1 indicates that we can compute the intersection number between X and Y if we are given generators in $\mathbf{T}_{\mathbb{Z}} \otimes \mathbb{Q}$ for $I_{(X,\mathbb{Q})}$ and $I_{(Y,\mathbb{Q})}$. In order to find generators in $\mathbf{T}_{\mathbb{Z}} \otimes \mathbb{Q}$ for $I_{(X,\mathbb{Q})}$, we compute a basis for $\mathbf{T}_{\mathbb{Z}} \otimes \mathbb{Q}$, say $\{t_1 \dots t_g\}$, and then solve the linear system in $c_1 \dots c_g$ over $\mathbb{Q}$ of all equations of the form $\sum_{i=1}^{g} c_i t_i x = 0$ as x varies over basis elements of X. Upon solving for the weights c_i, we compute the generators for $I_{(X,\mathbb{Q})}$ as $\sum_{i=1}^{g} c_i t_i$. Similarly, we can compute generators for $I_{Y,\mathbb{Q}}$.

In section 2.3.1, we describe how one may compute a basis for $T_{\mathbb{Z}} \otimes \mathbb{Q}$. In section 2.3.2, we show in more detail how one may solve the system of equations of the form $\sum_{i=1}^{g} c_i t_i x = 0$ for the weights c_i.

2.3.1 Computing a Basis for $\mathbf{T} \otimes \mathbb{Q}$

In SAGE, the matrix for Hecke Operator is given by the command matrix(). Because computing the matrix for a Hecke Operator is easy in SAGE, it is not difficult to check for linear dependancies between matrices for various Hecke operators until g (the dimension of $\mathbf{T} \otimes \mathbb{Q}$) have been collected.

Algorithm 1. *To compute a basis of matrices for the Hecke Algebra $\mathbf{T} \otimes \mathbb{Q}$:*

```
1. Initialize a list, denoted by S, containing only the matrix for $T_2$.
2. While $|S|$ is less than $g$:
    (a) Compute a matrix $M$ for a Hecke Operator that has not yet been checked for linear
        independence against the elements of $S$.
    (b) Choose at random, $|S|+1$ positions inside a $g \times g$ matrix and for each matrix $M_i$ in
        $S$ place the corresponding $|S|+1$ entries into a column vector $C_i$ and place the corre-
        sponding entries of $M$ into a column vector $C_{n+1}$.
    (c) Take the determinant of the matrix $[C_1 \dots C_{n+1}]$ (this can be done modulo a small prime).
    (d) If the determinant is nonzero, append $M$ to the list.
3. Output list.
```

2.3.2 Solving the Linear System

Let $s_1 \dots s_{2g}$ be a basis for $H_1(X_0(N),\mathbb{Z})$. In SAGE, the command X.modular_symbols() computes a basis for X_H whose elements are of the form $\sum_{i=1}^{2g} r_i s_i$. Let M be a matrix whose columns are of the form $\begin{bmatrix} r_1 \\ \cdot \\ \cdot \\ \cdot \\ r_{2g} \end{bmatrix}$, where $\sum_{i=1}^{2g} r_i s_i$ is a basis element of X_H. Then M gives a surjection M : $\mathrm{H}_1(X_0(N),\mathbb{Z}) \mapsto X_H$. For an $m \times n$ matrix A, denote by vector(A) the row vector of $m \cdot n$ entries constructed by horizontally augmenting the rows of A. Suppose $\{t_1 \dots t_g\}$ is a basis for $\mathbf{T} \otimes \mathbb{Q}$ with associated matrices $\{T_1 \dots T_g\}$ and suppose that a matrix $M : \mathrm{H}_1(X_0(N),\mathbb{Z}) \mapsto X_H$ gives a surjection. Then for some weights c_i, $\sum_{i=1}^{g} c_i t_{i|X} = 0_{|X}$ if and only if $\sum_{i=1}^{g} M(c_i t_i) = 0$, if and only if $\sum_{i=1}^{g} c_i M(t_i) = 0$, if and only if

$$\begin{bmatrix} c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_g \end{bmatrix} \in ker\left(\begin{bmatrix} \text{vector}(M(T_1)) \\ \cdot \\ \cdot \\ \cdot \\ \text{vector}(M(T_g)) \end{bmatrix}^{\mathrm{T}} \right). \tag{2.3.1}$$

Thus, in order to solve for generators of $I_{(X,\mathbb{Q})}$, we employ the following algorithm.

Algorithm 2. *To solve the linear system of equations $\sum_{i=1}^{g} c_i t_i x = 0$ as x varies over basis elements of X:*

1. *Construct M as defined above.*
2. *For each matrix T_i, multiply $M \cdot T_i$ and record the entries into a column vector C_i.*
3. *Compute the solution set $\left\{ \begin{bmatrix} c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_g \end{bmatrix} \right\}$ as $ker([C_1 \dots C_g])$.*
4. *Output matrices $\sum_{i=1}^{g} c_i T_i$.*

CHAPTER 3

COMPUTING A q-EXPANSION BASIS FOR $S_2(\Gamma_0(N))$ TO HIGH PRECISION

Let N be a positive integer. In this chapter, we describe a new method for computing a basis over $\mathbb{Q}$ of q-expansions for $S_2(\Gamma_0(N),\mathbb{Q})$ that performs better than other approaches when the number of terms computed in the expansion (precision) is high. The method we describe is based on computing matrices for prime-index Hecke operators T_p over irreducible Hecke-invariant subspaces of $S_2(\Gamma_0(N),\mathbb{Q})$.

In the introduction, we give the theoretical motivation for the algorithm, and in Section 3.2, we give an explicit description. In Section 3.3, we describe an algorithm for saturating a $\mathbb{Z}$-submodule that is particularly well suited for producing a $\mathbb{Z}$ basis for $S_2(\Gamma_0(N),\mathbb{Z})$ given a basis for $S_2(\Gamma_0(N),\mathbb{Q})$. In Section 3.4, we discuss some applications of the algorithms in Sections 3.2 and 3.3, namely: experimental computations with elliptic curves concerning cancellations in the conjectural Birch and Swinnerton-Dyer formula.

3.1 Introduction

Let M be the space of modular symbols for $\Gamma_0(N)$ invariant under the action of involution $i*$ as described in Section 1.1.2 and let m denote its dimension. Let S be the cuspidal modular symbols corresponding to the kernel of the boundary map described in 1.1.2, and let s denote its dimension. Let $\mathbf{N}$ be the modular symbols corresponding to the new subspace, and let n denote its dimension. (A basis for $\mathbf{N}$ can be computed as described in Section 1.1.2.)

Let G_f be the rational structure of a subspace of $S\otimes\mathbb{C}$ that corresponds (in the sense of Section 1.1.2) to the Galois orbit of a newform f (we will denote this orbit in $S_2(\Gamma_0(N),\mathbb{C})$ as $\bar{G}_f$), and let $d = deg(L/\mathbb{Q})$ where L is the normal closure of the number field generated by the coefficients of f. The subspace G_f is invariant under the action of $\mathbf{T}$, (see [6] page 39). We have a tower of inclusions $G_f \subseteq \mathbf{N} \subseteq S \subseteq M$ where $G_f, \mathbf{N}, S, M$ are of dimesnions d, n, s, and m respectively and all subspaces are Hecke-Invariant.

Proposition 4. *For any newform f, the Galois orbit $\bar{G}_f$ of f has a basis of forms with integer coefficients.*

Proof. See page 38 of [6] where this result is implied and used as the main ingredient of Theorem 1.31 in loc. cite. □

Proposition 5. *Let G be a Hecke-invariant subspace of $S_2(\Gamma_0(N),\mathbb{Q})$ and let B be a postive integer. Let $\{A_1 \dots A_B\}$ be matrices for the operators $\{T_1 \dots T_B\}$ in $\mathbf{T}_{|G}$ with respect to some basis for G. For any matrix entry $[ij]$, the power series $A_{1ij}q + A_{2ij}q^2 + \dots A_{Bij}q^B$ is the q-expansion of some form in G up to precision B.*

Proof. This is essentially a remark in page 306 of [9], only we change from all of $S_2(\Gamma_0(N),\mathbb{Q})$ to G where appropriate and add some context as we follow the model in loc cite. First consider the paring $G \times \mathbf{T}_{|G} \to \mathbb{Q}$ given by $(f,t) \mapsto a_1(t(f))$. To see that this pairing is perfect, we observe that if $(\cdot, f) := 0$ for some f, then since Proposition 5.3.1 in [7] implies $a_m(f) = a_1(T_m(f))$, we would have $(T_m, f) = a_m(f) = 0$ for all m, so $f = 0$. On the other hand, if $(t,\cdot) := 0$ for some $t \in \mathbf{T}_{|G}$, we would have $a_m(t(f)) = a_1(T_m(t(f))) = a_1(t(T_m(f))) = (t, T_m(f)) = 0$ for all m, so $t = 0$. Since the paring is perfect, any linear map in $\alpha \in Hom_{\mathbb{Q}}(\mathbf{T}_{|G},\mathbb{Q})$ can be given as $(\cdot, f)$ for some $f \in G$. Then $\alpha(T_m) = a_m(f)$ and $\sum_m \alpha(T_m)q^m = f$. To complete the proof to its current wording, we add only the remark that $\alpha : \mathbf{T}_{|G} \to \mathbb{Q}$ with $\alpha(t)$ mapping t to some fixed ij-th entry of t's matrix with respect to a given basis of G is indeed a linear map. □

While the above proposition shows how one can construct q-expansions from matrices for Hecke operators, the next theorem and its first corollary show that enough q-expansions can be constructed in this way to form a basis for all of G_f and give an explicit description of which d entries $[ij]$ can be used to guarantee linear independence among the constructed q-expansions.

(Remark: On a couple of occasions, as in the next theorem, it will be necessary to use the fact that there exists an operator t in $\mathbf{T}_{|\mathbf{N}} \otimes \mathbb{Q}$ such that t^i for $i \in [0, n-1]$ generate $\mathbf{T}_{|\mathbf{N}} \otimes \mathbb{C}$ as a vector space over $\mathbb{C}$. In particular, t has distinct eigenvalues. This is granted by the fact that the elements of $\mathbf{T}_{|\mathbf{N}} \otimes \mathbb{C}$ are diagonalized over a basis of newforms and form a $\mathbb{C}$-vector space of dimension n, and thus any vector in $\mathbb{C}^n$ will form the diagonal of some element of $\mathbf{T}_{|\mathbf{N}} \otimes \mathbb{C}$.)

Theorem 5. *Let g be a nonzero element of $\bar{G}_f$. The $\mathbf{T} \otimes \mathbb{C}$ orbit of g is all of $\bar{G}_f \otimes \mathbb{C}$.*

Proof. Recall that L denotes the normal closure of the number field containing the coefficients of f. Let $\{f_1 \dots f_d\}$ be the Galois orbit of f with $f = f_1$.

Step 1) Write g as $g = \sum c_i f_i$. We first prove that $c_i \neq 0$ for all i. Since $g \neq 0$, $c_n \neq 0$ for some n. Also, g is fixed by the action of $Gal(L/\mathbb{Q})$ since its coefficients are rational. Let k be an integer in $[1,d]$ and let $\sigma \in Gal(L/\mathbb{Q})$ be such that $\sigma(f_n) = f_k$. Then $\sigma(g) = g = \sum_{i \neq n} \sigma(c_i f_i) + \sigma(c_n f_n) = \sum_{i' \neq k} c_{i'} f_{i'} + \sigma(c_n) f_k$. If c_k were 0, then $c_n = \sigma^{-1}(c_k)$ would be too. Thus all weights c_i are nonzero.

Step 2) Let t be some element of $\mathbf{T} \otimes \mathbb{C}$ with distinct eigenvalues, and for any i as above, denote by λ_{f_i} t's eigenvalue corresponding to f_i. Then define $\tau_i \in \mathbf{T} \otimes \mathbb{C}$ by $\tau_i = \prod_{k \neq i}(t - \lambda_{f_k})$. Then the set $\{\tau_1(g) \dots \tau_d(g)\}$ forms a basis for $\bar{G}_f$. This concludes the proof.

□

Corollary 1. *Let $\{t_1 \dots t_d\}$ be any basis of matrices with rational entries for $\mathbf{T}_{|G_f}$ with respect to some rational basis $\{g_1 \dots g_d\}$ for $\overline{G_f}$. Keeping $j : 1 \leq j \leq d$ fixed, the j-th rows of these t_i are linearly independent. In particular, the first rows of these t_i are linearly independent.*

Proof. Suppose the corollary didn't hold, then the dimension of the $\mathbf{T} \otimes \mathbb{C}$ orbit of g_j would be less than d. This concludes the proof. □

The above theorem shows that if we can compute matrices for the Hecke operators $T_1 \dots T_B$ with respect to some basis of $\mathbf{T}_{|G_f}$, then we can take the first row of each to construct a q-expansion basis for G_f to precision B. The next section addresses in detail how to compute the matrices T_n in $\mathbf{T}_{|G_f}$. We end the introduction with a second corollary to Theorem 5.

Corollary 2. *The spaces $\bar{G_f}$ and G_f are **T**-simple.*

Proof. That $\bar{G_f}$ is $\mathbf{T}$-simple follows directly from the theorem. That G_f is $\mathbf{T}$-simple follows since it is the image of $\bar{G_f}$ under the $\mathbf{T}$ compatible isomorphism described in Section 1.1.2. $\square$

3.2 Matrices for Operators on Invariant Subspaces

Let M be a $\mathbb{Q}$ vector space of dimension m and let T be a space of operators acting on M. Let $G \subseteq M$ be a subspace of dimension $d << m$ that is invariant under the action of T and suppose $T_{|G}$ is also of dimension d. Suppose M has a basis $\{\alpha_1 \dots \alpha_m\}$. Suppose the action of any given operator $t \in T$ on an element of M can only be determined by t's action on these basis elements. The subspace G has a basis whose elements are of the form $\beta = \sum a_j \alpha_j$, thus computing the action of t on a general element in G requires approximately m times more computation than computing t's action on a basis element α_j.

Suppose that we want matrices for the action of operators t only on a basis of G, as opposed to all of M. Since G is smaller, is there a way to compute matrices for t on G that is less computationally costly than computing the matrix for t on M and then restricting to G?

The naive approach of recording t's action on the basis for G and building the matrix accordingly would be a very inefficient method. Since t's action on a linear combination is determined by its action on the summands, this method would be no less costly than building a matrix for t over all of M. Below we describe an approach for determining t by its action on fewer basis elements α_j.

Suppose that we can somehow efficiently compute a basis $\{t_1 \dots t_d\}$ for $T_{|G}$. Then if we could then compute $t(\beta)$ for enough $\beta \in G$, we could solve the system equations of the form $\sum c_i t_i(\beta) = t(\beta)$ and get t as a linear combination of t_i. So then what would be ideal is if there were several β in G that were either themselves basis elements α_j or linear combinations of very few of the α_j, since the action of t on such β would be easier to compute. This is highly unlikely because $d << m$.

However, it may be possible to find a subset Ω of the basis elements α_j such that t is specified by its action on the elements of Ω. If we could compute $\text{proj}_G t_i(\alpha_j)$ for all t_i in the basis for $T_{|G}$ and all $\alpha_j \in \Omega$ and solve the system $\sum c_i \text{proj}_G t_i(\alpha_j) = \text{proj}_G t(\alpha_j)$ for the weights c_i, then we would have $t = \sum c_i t_i$. The observatation is that because G is invariant under the action of t, $\text{proj}_G t(\alpha) = t(\text{proj}_G(\alpha))$, where the term on the left is easy to compute and the term on the right is hard to compute.

3.2.1 Example: Matrices for Hecke Operators on the New Subspace when the Old and Eisenstein Subspaces Dominate

Now let M be the space of modular symbols fixed under involution for $\Gamma_0(N)$ and let $\mathbf{N}$ be the new subspace of M. Denote by π the projection map from M onto $\mathbf{N}$. (Most intuitively, if d denotes a degeneracy map from a lower level and d' denotes its inverse, the map π on a symbol s could be computed as $\pi : s \mapsto s - \sum_{d|N} d(d'(s))$ (read as: "s minus any old part"), but we describe in the next section another approach that suffices.)

In this example, consider $N = 500$. We compute T_{1511} for some basis of $S_2(\Gamma_0(500), \mathbb{Q})$. At level 500, $m = 78$; $n = 8$. Clearly the Eisenstein and old subspaces dominate M. $T_{3|\mathbf{N}}$ has distinct eigenvalues, so $T_3^0 \ldots T_3^7$ generate $T_{|\mathbf{N}}$. So to compute a matrix for T_{1511}, we find several basis Manin symbos with nonzero projections onto **N**- in this case $(1,0), (1,100)$ and $(4,131)$ suffice- and set up the matrix

$$A = \begin{bmatrix} \pi(T_3^0((1,0))) & \pi(T_3^2((1,0))) & \ldots & \pi(T_3^7((1,0))) \\ \pi(T_3^0((1,100))) & \pi(T_3^2((1,100)) & \ldots & \pi(T_3^7((1,100))) \\ \pi(T_3^0((4,131))) & \pi(T_3^2((4,131))) & \ldots & \pi(T_3^7((4,131))) \end{bmatrix}$$

and the vector

$$b = \begin{bmatrix} \pi(T_{1511}((1,0))) \\ \pi(T_{1511}((1,100))) \\ \pi(T_{1511}((4,131))) \end{bmatrix}.$$

Note that the matrix A has 21 rows and 7 columns, because each entry in the expression above represents a vector of length 7. The solution to $Ax = b$ is $c = [110/3, 0, -677/6, 0, 295/6, 0, -23/6, 0]$. We compute T_{1511} as $\sum c_i T_3^i$ and find

$$T_{1511|\mathbf{N}} = \begin{bmatrix} 10 & -71/2 & 129/10 & -99/20 & -47/5 & -11/5 & 177/5 & -53/5 \\ -2 & 101/2 & -261/10 & 281/20 & 23/5 & 29/5 & 97/5 & -168/5 \\ 10 & -23/2 & 23/2 & -3/4 & -13 & 15 & 5 & 12 \\ 0 & -8 & 0 & 52 & 0 & -8 & 8 & -8 \\ -9 & -23/2 & -1/10 & 1091/20 & 118/5 & -196/5 & 82/5 & -148/5 \\ 4 & -47/2 & 229/10 & -189/20 & -27/5 & 94/5 & -133/5 & 222/5 \\ -2 & 49/2 & -333/10 & 1203/20 & 9/5 & -58/5 & 341/5 & -359/5 \\ -4 & 51/2 & -301/10 & 1271/20 & 13/5 & -41/5 & 237/5 & -273/5 \end{bmatrix}$$

One could of course compute the matrix above by taking the approach of [13], [18], and [19] to compute T_{1511} acting on the full basis for M and restricting to the basis of **N**. However, this requires 78 computations of T_{1511} acting on a Manin symbol whereas the approach above only required 3 such computations (after our matrix A has been computed).

3.2.2 Restricting to Irreducible T-Invariant Subspaces G_f

The restriction to **N** in Example 2.1 is useful if the old and Eisenstien subspaces dominate the space of modular symbols, but it will offer no real improvement at, for instance, prime levels - where the space is almost entirely new already. But the newspace can be further decomposed into subspaces G_f.

Determining Matrices t in $T_{|G_f}$. In Example 3.2.1, the matrix A had 21 rows and only 7 columns, so the system $Ax = b$ is over-determined. This was because **N** in the example was not an irreducible **T**-simple subspace, so the action of T_p on a single Manin symbol in M may not completely determine its matrix on a rational basis of **N** (see the remark below Proposition 6). However, when we restrict to G_f, it is only necessary to know the action of T_p on a single Manin symbol.

Proposition 6. *Suppose the Manin symbol s has nonzero projection onto $G_{|f}$. An operator t in $\mathbf{T}_{|G_f} \otimes \mathbb{Q}$ is completely determined by t's action on s.*

Proof. Let B be a $\mathbf{T}_{|G_f} \otimes \mathbb{C}$-diagonalizing basis for $G_f \otimes \mathbb{C}$. Since G_f is $\mathbf{T}$-simple, it contains no nonzero proper $\mathbf{T}$ submodules. It follows that the $\mathbb{C}$ valued coordinate vector for s with respect B must contain either all zero entries or no zero entries because otherwise, the $\mathbf{T}$ orbit of s would be a nonzero proper $\mathbf{T}$ submodule of G_f.

Now suppose $t(s) = 0$. If t is diagonalized with respect to B and s is given as a coordinate vector with respect to B, then $t(s) = 0$ implies that t acts as 0 on each element of B. Thus $t := 0$. This concludes the proof. □

(Remark: More generally, operators $t \in \mathbf{T}_{|\mathbf{N}}$ will also be determined by their actions on a single Manin symbol as long as the symbol's projection onto each invariant subspace G_f is nonzero. In the worst case, t must be determined by its action on $\#(\{G_f\})$ symbols, where $\#(\{G_f)\}$ denotes the number of irreducible $\mathbf{T}$-simple subspaces of $\mathbf{N}$.

Although it would have been impossible to know without computing projections onto the irreducible $\mathbf{T}$-simple subspaces, the symbol $(4,131)$ was not needed in Example 3.2.1 since the sum $(1,100)+(1,300)$ has nonzero projections onto all subspaces G_f)

Computing Projections $\pi_f : M \to G_f$. In order to carry out the method in Example 3.2.1 on smaller spaces G_f, we also need projection maps $\pi_f : M \to G_f$. The following method was implied in [18] and described in limited detail in [19] . Here we describe the idea again and add some context. Algorithm 7.17 in Chapter 7 of [18] shows how to decompose a vector space into t-simple spaces, where t is an operator acting on the space. In our context, if the Hecke operator t has distinct eigenvalues, then in view of Corollary 2 and the remark above Theorem 5, the algorithm will decompose the newspace $\mathbf{N}$ into $\oplus_f G_f$. Call M_{G_f} the matrix whose rows are basis elements for G_f in terms of basis elements for M. And call M_O and M_E matrices whose rows are basis elements for the old and Eisenstein spaces respectively. The matrix *mat* whose columns are the M_{G_f} augmented together, along with M_O and M_E on the right gives a map $mat : \oplus_f G_f \oplus O \oplus E \to M$ where O and E are the old space and Eisenstein spaces respectively. So mat^{-1} gives a map $mat^{-1} : M \to \oplus_f G_f \oplus O \oplus E$. Then taking the rows in mat^{-1} corresponding to G_f gives a projection $\pi_f : M \to G_f$.

Revisiting the example above at level 500, we compute a block diagonal form of T_{1511} by solving for its blocks at each invariant subspace G_f.

There are 3 irreducible $\mathbf{T}$-simple subspaces G_f of $\mathbf{N}$. Call these $G_{f_1}, G_{f_2}, G_{f_3}$. The symbol $(1,100)$ has nonzero projection onto G_{f_1} and G_{f_3}, and $(1,300)$ has nonzero projection onto G_{f_2}. The subspace G_{f_1} has dimension 2, thus $T_3{}^0_{|G_{f_1}}$ and $T_3{}^1_{|G_{f_1}}$ generate its Hecke algebra. We set up the matrix

$$A = \begin{bmatrix} \pi_{f_1}(T_3{}^0(1,100))) & \pi_{f_1}(T_3{}^1(1,100))) \end{bmatrix}$$

and the vector

$$b = \begin{bmatrix} \pi_{f_1}(T_{1511}(1,100))) \end{bmatrix}$$

And solve $Ax = b$ for $c = [3,4]$ and compute $T_{1511_{G_{f_1}}} = 3 * T_3{}^0_{|G_{f_1}} + 4 * T_3{}^1_{|G_{f_1}} = \begin{bmatrix} -5 & -4 \\ 20 & 15 \end{bmatrix}$

Again, this is the exact matrix that could be computed as the matrix for T_{1511} over a basis for M restricted to G_{f_1}, but this approach would require knowing T_{1511}'s action on 78 Manin symbols whereas in our example we only needed T_{1511}'s action on a single Manin symbol. Carrying through the same way with G_{f_2} and G_{f_3} we compute block diagonal matrix of $T_{1511|\mathbf{N}}$ as

$$T_{1511|\mathbf{N}} = \begin{bmatrix} -5 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -5 & -12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20/3 & 15 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 36 & -32 & 8 & 32 \\ 0 & 0 & 0 & 0 & 0 & 28 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & -32 & 44 & 0 \\ 0 & 0 & 0 & 0 & 0 & -8 & 0 & 52 \end{bmatrix}$$

3.2.3 The Algorithm

Since for each f, G_f has a basis of forms with rational coefficients, in order to get a basis of such forms for $\mathbf{N}$, one can compute projections $\pi_f : M \to G_f$ for each newform $f \in \mathbf{N}$ and then set up the system $Ax = b$ as in the example above and solve for enough matrices $T_{p|G_f}$ to get a q-expansion basis for G_f for every f and simply append all of these together for a q-expansion basis for $\mathbf{N}$. If one wants a q-expansion basis for S, simply do as above at each level dividing the level of S and level lift through upward degeneracy maps.

Remark: If G_f is one-dimensional, we prefer to compute prime-index coefficients of f by counting points on the associated elliptic curve over prime fields. This can be done in polynomial time with the Schoof-Elkies-Atkin algorithm, but is in practice done with the Shanks-Mestre baby-step/giant-step method or an explicit formula based on Cornacchia's algorithm as described in [16]. Not only is this approach faster, but it also may lower the total number of modular symbols needed for the computation (if, for instance, the symbols used to set up equations on other Galois orbits all have zero projections onto f).

This discussion leads to:

Algorithm 3. *In order to compute a q-expansion basis for the new subspace of $S_2(\Gamma_0(N), \mathbb{Q})$ up to precision B,*

1. *Search for $t \in \mathbf{T}$ such that $t_{|N}$ has distinct eigenvalues as described in Section A.1 of the appendix.*

2. *As described in Section 3.2.2 above, decompose the $\mathbf{N}$ into subspaces that are t-simple, i.e., decompose $\mathbf{N}$ as $\oplus G_f$ as f ranges over newforms in $\mathbf{N}$.*

3. *As described in Section 3.2.2, compute projection maps π_f onto all t-simple subspaces G_f computed in Step (2) that have dimension greater than 1.*

4. *For each t-simple subspace G_f in Step (2):*

   (a) For $i \in [0, d-1]$ where $d = dim(G_f)$, compute the restrictions of t^i onto G_f and save as a basis for $\mathbf{T}_{|G_f}$.

   (b) Find a symbol s_f with nonzero projection onto G_f.

   (c) Set up the matrix $A_{G_f} = \begin{bmatrix} \pi_f(t^0(s_f)) & \pi_f(t^1(s_f)) & \ldots & \pi_f(t^{d-1}(s_f)) \end{bmatrix}$

5. *At all primes p up to B:*

(a) *For each t-simple subspace G_f:*

i. *Compute the vector $b_{G_f,p} = \left[\pi_f(T_p(s_f))\right]$*

ii. *Solve the system $A_{G_f}x = b_{G_f,p}$ for weights $c_0 \dots c_{d-1}$ and compute the matrix $T_{p|G_f} = \sum c_i t^i$.*

6. *Compute the matrices for T_n for composite n using the multiplicative formulas for Hecke operators given in Section 1.1.1.*

7. *For each index $i \in [1,B]$, concatinate the first rows of the matrices $T_{i|G_f}$ as f ranges over conjugacy classes of newforms with $dim(G_f) > 1$ and save as a list C_i of length $r = \sum_{G_f|dim(G_f)>1} dim(G_f)$.*

8. *For each 1-dimensional subspace G_f, compute the coefficients a_n for $n \leq B$ in the L-function of the elliptic curve associated to f and construct the corresponding q-expansions.*

9. *Output the q-expansions from Step (8) along with all q-expansions $\sum_i (C_i)_j q^i$ as j varies over $1 \dots r$.*

Remark: Step (6) in the description above occurs on matrices of size $d \times d$, where $d = dim(G_f)$. This means that aside from reducing the cost of computing prime-index Hecke matrices over G from m computations of T_p acting on a Manin symbols to at most $\#(\{G_f\})$, we also reduce the cost of matrix multiplication from $O(m^\gamma)$ to $O(\#(\{G_f\})max(dim(G_f))^\gamma)$ where γ depends on the multiplication algorithm used. The memory cost of storing the matrix also reduces from $O(n^2)$ to $O(\#(\{G_f\})max(dim(G_f))^2)$. In many cases, this allows for successful executions of computations that would otherwise fail because of memory exhaustion.

Given q-expansion bases for new subspaces at levels dividing N, it is not involved to compute a full q-expansion basis for all cusp forms at level N.

Algorithm 4. *In order to compute a q-expansion basis for $S_2(\Gamma_0(N), \mathbb{Q})$,*

1. *For every divisor D of N:*

   (a) *Compute a q-expansion for the new subspace at level D as in Algorithm 3 above*

   (b) *Map the q-expansions computed in step (a) by the appropriate upward degeneracy maps to $S_2(\Gamma_0(N))$*

2. *Output all degeneracy images computed in step (b) above*

Table 3.1: Tests at composite levels.

Level	B	Using SAGE's _q_expansion_module_rational()	Using Algorithm 4
250	3000	357.809	38.60
500	3000	1557.542	77.315
1000	3000	No answer produced; memory exhausted after 2100 s	188.3

Table 3.2: Tests at prime levels.

Level	B	Using SAGE's _q_expansion_module_rational()	Using Algorithm 4
211	3000	94.196	21.618
601	3000	514.524	98.6
809	3000	No answer produced; memory exhausted after 600 s	289.9

Remark: At prime levels, the method we describe is not as significant of an improvement in general. The cuspidal subspace at prime levels will often decompose into very large orbits where solving the corresponding systems becomes costly, and we no longer enjoy the advantage of throwing out the old subspace when we restrict.

Table 3.3: Tests at levels where newforms with integer coefficients dominate the new subspace and we compute a large portion of coefficients by counting points on the associated elliptic curves mod p.

Level	B	Using SAGE's _q_expansion_module_rational()	Using Algorithm 4
100	3000	89.328	1.044
200	3000	286.483	2.166
550	3000	No answer produced; memory exhausted after 1620 s	70.753
702	3000	No answer produced; memory exhausted after 3420 s	56.944
960	3000	No answer produced; memory exhausted after 2160 s	51.358

3.3 Local Saturation

In this section, we address the problem of finding a basis for $S_2(\Gamma_0(N),\mathbb{Z})$ given a basis for $S_2(\Gamma_0(N),\mathbb{Q})$.

Definition 13. *For any $\mathbb{Z}$ module $M \subset \mathbb{Z}^n$, we denote by $sat(M)$ the submodule $sat(M) = \{v \in \mathbb{Z}^n | d \cdot v \in M \text{ for some } d > 0\}$ and call it the **saturation of** M. For a prime p, we denote by $sat_p(M)$ the submodule $sat_p(M) = \{v \in \mathbb{Z}^n | p^a \cdot v \in M \text{ for some } a \geq 0\}$ and call it the **local saturation of** M **at** p. For a prime p, we denote by $partsat_p(M)$ the submodule $partsat_p(M) = \{v \in \mathbb{Z}^n | p \cdot v \in M\}$ and call it the **partial local saturation of** M **at** p. We also extend these definitions to integer matrices M, so that $sat(M)$, $sat_p(M)$, and $partsat_p(M)$ are matrices that generate the corresponding $\mathbb{Z}$ modules related to the rowspace of M, which we denote $rs(M)$.*

Clearing denominators in a rational q-expansion basis gives a submodule M of $S_2(\Gamma_0(N),\mathbb{Z})$ whose saturation is the entire module. It was observed in [19] that composite operation $\circ_{p_i}(sat_{p_i}(M))$ over all primes p_i dividing $[M : sat(M)]$ produces $sat(M)$. However, this method has not been used in practice because saturating locally was not efficient enough to outcompete saturation over $\mathbb{Z}$. In the rest of this section, we describe a more efficient method for saturating locally that makes finding the full $\mathbb{Z}$ saturation through combining local saturations feasible.

Let M be an integer matrix. Since $rs(M) \subseteq rs(partsat_p(M)) \subseteq rs(sat_p(M))$, with $rs(M) = rs(sat_p(M))$ only if $rs(M) = rs(partsat_p(M))$, the local saturation $sat_p(M)$ can be computed as the recursive limit of partial local saturations $partsat_p(M)$.

Definition 14. *Let M be a matrix in $Mat(\mathbb{Z})$, and let p be a prime. Let $\overline{M}$ denote the image of M in $Mat(\mathbb{Z}/p\mathbb{Z})$. Let $\overline{K}$ be the reduced echelon form of a left-kernel of $\overline{M}$, that is- let $\overline{K}$ be the matrix in reduced echelon form whose columns form a basis of the kernel of $\overline{K}^T$. We define the* mod p kernel of M to be the unique matrix with entries in $(-\frac{p}{2}, \frac{p}{2}]$ formed by lifting $\overline{K}$ to $Mat(\mathbb{Z})$ and we denote it as K.

By definition $rs(partsat_p(M)) \neq rs(M)$ only if there exist $w \in rs(partsat_p(M))$ such that $w \notin rs(M)$ while $p \cdot w \in rs(M)$. Let W be the $\mathbb{Z}$-span of all such w, then we proceed to compute $rs(partsat_p(M))$ as $rs(M) \cup W$.

Since all w in a basis for W are of the form K_iM/p, we have:

Proposition 7. *Let M be a matrix with entries in $\mathbb{Z}$ with mod p kernel K. Let be $l(K_i)$ the index of the leading 1 in K's i-th row. Let M' be the matrix M with its l-th row is replaced by $w = K_iM/p$ if $l(K_i) = l$. Then M' is a partial local saturation at p.*

Proof. It's obvious by construction that $W \subseteq rs(M')$. That $rs(M) \subseteq rs(M')$ follows from the following. (Recall that K was defined to be in reduced echelon form; otherwise the following does not work.) If M_i is a row in M that remains a row in M' then clearly M_i lies in the rowspace of M'. On the other hand, if M_l is a row in M with $l = l(K_i)$ for some i, then $M_l = M'_l - \sum_{j \neq l} K_iM_j = M'_l - \sum_{j \neq l} K_iM'_j$. □

This discussion leads to:

Algorithm 5. *In order to compute the local saturation of a matrix M at the prime p:*

```
1. Compute the mod p kernel matrix $K$ of $M$.
2. If $rank(K) > 0$, compute $partsat_p(M)$ by:
    (a) For each row $K_i$ in $K$:
        i. Compute $l(K_i)$, the position of $K_i$'s leading 1.
        ii. Replace $M_l$ with $K_iM/p$
3. To compute $sat_p(M)$, return to step (1).
4. Output the matrix $M$
```

Example. Let $M = \begin{bmatrix} 5 & 6 & 7 \\ 4 & 3 & 2 \\ 1 & 0 & 2 \end{bmatrix}$. The vectors $[6,6,6]$ and $[6,6,9]$ lie in $rs(sat_3(M))$ but do not lie in $rs(M)$. Thus $M \neq partsat_3(M)$. In order to compute $partsat_3(M)$, we find the mod 3 echelonized kernel matrix of M $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \in M_2(\mathbb{Z}/3\mathbb{Z})$. and lift the elements to $\mathbb{Z}$ to get $K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \in M_2(\mathbb{Z})$. Since $K_1 M = [6,6,9]$ and $l(K_1 = 1$, we replace the first row of M with $[6,6,9]/3$. Likewise, since $K_2(M) = [3,3,0]$ and $l(K_2 = 2$, we replace the second row of M with $[3,3,0]/3$. We have $M' = \begin{bmatrix} 2 & 2 & 3 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix}$. Since the mod 3 kernel of M' is of rank 0, M' is the local saturation of M at 3.

Computing $sat(M)$. Given a method for computing a local saturation $sat_p(M)$ like the one above, it is not involved to compute the full saturation.

Algorithm 6. *To compute the full saturation of a matrix M:*

1. *Compute a multiplicative bound Ω for $[M : sat(M)]$ as described in Section A.2 of the appendix.*
2. *For a prime $p|\Omega$:*
    (a) *Replace M by $sat_p(M)$.*
3. *Repeat step (2) until all primes dividing Ω have been exhausted.*

Table 3.4: Saturating a $\mathbb{Q}$ basis for $S_2(\Gamma_0(N),\mathbb{Q})$ to compute a basis for $S_2(\Gamma_0(N),\mathbb{Z})$ at levels 500 and 1000 at varying precisions.

Level	B	Using SAGE's saturation()	Using Algorithm 6
500	200	7.466	2.026
500	1000	14.11	8.013
500	3000	32.576	26.083
1000	200	6449.512	26.046
1000	1000	6319.492	74.981
1000	3000	6563.078	211.521

Remark: SAGE's saturation() strongly favors modules that are already highly saturated. The tests above are perfomred on q-expansion bases produced by Algorith 4 described in Section 2 of this chapter, and that algorithm tends to produce bases that are highly saturated- especially when the cuspidal space is dominated by 1-dimensional Galois orbits or degeneracy images of them, since our method already saturates these subspaces. The following table illustrates this pre-conditioning advantage of Algorithm.

Table 3.5: Saturating $\mathbb{Q}$ bases for $S_2(\Gamma_0(N),\mathbb{Q})$ computed with SAGE's saturation() and using Algorithm 6

Basis for $S_2(\Gamma_0(211)$ Computed Using	B	Using SAGE's saturation()	Using Algorithm 6
Algorithm 4	1000	3.888	2.732
_q_expansion_module_rational()	1000	379.715	1.491

3.3.1 Computing Integral Bases of q-Expansions

Using Algorithm 6 on the output of Algorithm 4 produces an integral basis of q-Expansions for $S_2(\Gamma_0(N,\mathbb{Z}))$.

Algorithm 7. *To compute an integral basis of q-expansions for* $S_2(\Gamma_0(N,\mathbb{Z}))$*:*

1. *Compute a rational basis of q-expansions for* $S_2(\Gamma_0(N,\mathbb{Z}))$.

2. *Clear denominators in the output from step (1) and saturate using Algorithm 6.*

3. *Output saturated basis*

Table 3.6: Computing an integral basis for $S_2(\Gamma_0(N))$

Level	B	Using Sagee's _q_expansion_module_integral()	Using Algorithm 7
211	1000	511.61	9.30
211	2000	3344.63	19.97
500	1000	505.60	38.92
500	2000	1013.32	72.86
1000	1000	2601.09	155.07
1000	2000	memory exhausted	282.30

3.4 Applications

In this section, we discuss a couple of applications of the algorithms in sections (2) and (3) and present experimental data for congruence primes and elliptic curves that give evidence for Agashe's conjectures [aga] about cancellations in the conjectural Birch and Swinnerton-Dyer formula.

3.4.1 Computing Congruence Primes

Throughout this section, let X and Y be subspaces of $S_2(\Gamma_0(N))$. Recall that we call a prime p a *congruence prime* between X and Y if there exists some cusp forms f in X and g in Y with integer Fourier coefficients such that $a_n(f) \equiv a_n(g)$ mod p for all n, (in which case, we write $f \equiv g$ mod p). Congruence primes play an important role in number theory. For instance, they were used extensively in the work of Ribet [15] and Wiles [22], among others, leading to the proof of Fermat's Last Theorem.

Deciding if a Given Prime is a Congruence Prime. Here we give a criterion for deciding if a given prime p is a congruence prime between X and Y.

Let

$$B = \left\lfloor \frac{[SL_2(\mathbb{Z}) : \Gamma_0(N)]}{6} - \frac{[SL_2(\mathbb{Z}) : \Gamma_0(N)] - 1}{N} \right\rfloor. \tag{3.4.1}$$

The following is a consequence of Theorem 1 in [20].

Theorem 6 (Sturm). *Let h and g be two cusp forms in $S_2(\Gamma_0(N))$ and let p be a prime. If $a_n(h) \equiv a_n(g) \bmod p$ for all n up to B then $a_n(h) \equiv a_n(g) \bmod p$ for all n.*

The integer B above is called the *Sturm bound* of $\Gamma_0(N)$. If g is a cusp form, then let $v(g)$ denote the row vector whose components are the first B coefficients of g. Let $\{g_1, \ldots, g_r\}$ be a set of cusp forms with integer Fourier coefficients whose $\mathbb{Z}$-span is $X \cap S_2(\Gamma_0(N), \mathbb{Z}))$. (Such a set will necessarily exist by Proposition 1 as long since X is of the form .) Let M_X be the integer matrix whose i-th row is $v(g_i)$ for i=1, ..., r. Let M_Y be defined analogously with X replaced by Y in the above. Let M_{XY} denote the vertical concantination of M_X and M_Y with M_X stacked on top of M_Y Denote by $rank_p(M)$, the mod p rank of a matrix.

Proposition 8. *There exists mod p congruence linking the spaces X and Y if and only if $rank_p(M_{XY}) < rank_p(M_X) + rank_p(M_Y)$.*

Proof. A decrease in rank implies a row in X is a mod p linear combination of the rows in Y, i.e. the two rows are congruent mod p. ☐

In the event that Y is of rank 1 (e.g. a newform f with integer coefficients), there is an easy way to find all congruence primes between the newform $Y = f$ and X.

Let E be the elliptic curve over $\mathbb{Q}$ associated to f. Consider the map $\phi : X_0(N) \to J_0(N)$ given by $\phi : P \mapsto (P) - (\infty)$. The map ϕ induces a surjective morphism

$$X_0(N) \to J_0(N) \to E.$$

The degree of this composite morphism is called the *modular degree* of E.

The following is Theorem 3.11 in [3].

Proposition 9 (Agashe, Ribet, Stein). *Suppose f is a newform of level N with associated elliptic curve E, and f is congruent modulo p to a cusp form in its orthogonal complement in $S_2(\Gamma_0(N))$. Then either p divides the modular degree of E or $p^2|N$.*

The modular degree can be computed very efficiently. For a history of computational approaches of calculating the modular degree, see the introduction in [21]. In SAGE, Watkins's algorithm [21] for computing the modular degree of an elliptic curve is called by modular_degree().

Proposition 8 provides an efficient test to check if a prime is a congruence prime between f and the X, while Proposition 9 gives a finite list of candidate primes that contains all congruence primes. Together, we have

Algorithm 8. *Given a newform f with integer coefficients and a spanning set for X of forms with integer coefficients, this algorithm computes all congruence primes between f and X.*

1. *Compute the matrix M_{Xf} as described above.*

2. *Compute the modular degree and generate finite list of primes $\{p_1 ... p_n\}$ such that p_i divides the modular degree or $p_i^2 | N$. (Recall that by Proposition 9, these are the only primes that could be congruence primes between f and X.)*

3. *For each p_i, check if the $v(f)$ is in the row space of M modulo p_i.*

4. *Output the list of primes for which the answer in step three is yes.*

3.4.2 Cancellations in the Conjectural Birch and Swinnerton-Dyer Formula

Let $E/\mathbb{Q}$ be a modular elliptic curve associated to the newform f of level N. Recall the conjectural Birch and Swinnerton-Dyer Formula:

$$\frac{L^r(E/\mathbb{Q},1)}{r!\,\Omega} = \frac{|\text{Ш}(E/\mathbb{Q})| \cdot \mathrm{R}(E/\mathbb{Q}) \cdot \prod_p c_p}{|E_{\text{tors}}(\mathbb{Q})|^2}. \tag{3.4.2}$$

The right hand side of equation 3.4.2 has been studied extensively (see the introduction in [11]). In particular, in regard to the relationship between the Tamagawa product $\Pi_p c_p$ and the order of $E_{\text{tors}}(\mathbb{Q})$, M. Emerton [8] showed that when N is prime, $\Pi_p c_p = |E_{\text{tors}}(\mathbb{Q})|$.

When N is not prime, $|E_{\text{tors}}(\mathbb{Q})|$ need not equal $\Pi_p c_p$. In fact, this first occurs when $N = 42$. D. Lorenzini [11] has shown that if $\ell > 3$ is a prime number, then the order of the ℓ-primary part of $E_{\text{tors}}(\mathbb{Q})$ divides $\Pi_p c_p$.

In the other direction, Agashe [1] has shown that:

Proposition 10. *Let ℓ be an odd prime such that either $\ell \nmid N$ or for all primes r that divide N, $\ell \nmid (r-1)$. If ℓ divides the order of the geometric component group of E at p for some prime $p||N$, then either $E[\ell]$ is reducible or the newform f is congruent to a newform of level dividig N/p (for all Fourier coefficients whose indices are coprime to $N\ell$) modulo a prime ideal over ℓ in a number field containing the Fourier coefficients of both newforms.*

In fact, in [1] the author conjectures more specifically that:

Conjecture 4 (Agashe)**.** *If an odd prime ℓ divides c_p for some prime p, then either ℓ divides the order of $E_{\text{tors}}(\mathbb{Q})$ or the newform f is congruent modulo ℓ to a form in the subspace generated by degeneracy map images of newforms of levels dividing N/p.*

Agashe was able to test Conjecture 4 through the first 2463 optimal elliptic curves (up to conductor 1000), before computations became too slow to be feasible. Using Algorithms 7 and 8 we tested Conjecture 4 through the first 9515 optimal elliptic curves (up to conductor 3000).

Theorem 7. *Conjecture 4 holds for all optimal elliptic curves with conductor less than or equal to 3000.*

.1 Finding an element of $\mathbf{T}_{|\mathbf{N}}$ with distinct eigenvalues

Our approach is only heuristic, but has never in practice failed to work. We test T_2 through T_{13} to see if any of these matrices have distinct eigenvalues. If one of these do have distinct eigenvalues, then we're done. If not, then we take $\sum 1/nT_n$. Heuristically, one should expect this approach to fail only if two newforms have the same first 13 coefficients.

.2 Computing a multiplicative bound Ω for $[sat(M):M]$

A prime divides $[sat(M):M]$ only if it divides the determinant of a submatrix M' of M. This follows from the fact that $p|[sat(M):M]$ implies that there is a p-linear dependence between the rows of M which restricts to the submatrix M', thus the determinant of M' is 0 modulo p. A naive approach then is to compute Ω by taking the gcd of determinants of several submatrices. However, in practice, these determinants may often be 0 because M' may be very sparce. What can be done instead is to construct a matrix $\bar{M}$ whose columns are random linear combinations of columns of M. This makes sparcity of $\bar{M}$ less likely, so that $det(\bar{M})$ is more likely to be nonzero. The gcd of several determinants of matrices $\bar{M}$ constructed thusly suffices as a bound Ω.

.3 Code

.3.1 Algorithms 1 and 2

Generating the $\mathbf{T}\otimes\mathbb{Q}$ Basis.

```
def Basis_For_TQ(N,g):
    ##N=S.ambient().cuspidal_submodule()
    B=[N.T(2).matrix()]
    _B=[N.T(2)]
    length=1
    n=3
    entries=[[ZZ.random_element(g),ZZ.random_element(g)]]
    rows=[[B[0][entries[0][0]][entries[0][1]]]]
    added_new_row=true
    while len(B)<g:
        _C=N.T(n)
        C=_C.matrix()
        newest_entry=[ZZ.random_element(g),ZZ.random_element(g)]
        if added_new_row:
            entries.append(newest_entry)
            for i in range(0,len(B)):
                rows[i].append(B[i][newest_entry[0]][newest_entry[1]])
        last_row=[]
        ##note that this is enough since this will only produce false negatives
        for j in range(0, len(entries)):
            ##this is for the candidate matrix
            last_row.append(C[entries[j][0]][entries[j][1]])
        rows.append(last_row)
        if Matrix(GF(7),rows).determinant()!=0:
             B.append(C)
             _B.append(_C)
             added_new_row=true
        else:
            rows.pop()
            added_new_row=false
            entries.pop()
            entries.append(newest_entry)
        n=n+1
        print n
    return [B,_B]
```

Generating the Basis Transformation Matrix.

```
def Find_Basis_Transformation(N,X,g):
    first_term_list=[]
    basis=N.basis()
    Transformation=[]
    for i in range(0,len(X.basis())):
        column=[]
        for j in range(0,g):
            column.append(0)
        Transformation.append(column)
    for i in range(0,len(basis)):
        s=str(basis[i])
        first_term_list.append(s[0:s.find(')')]+')')
    for i in range(0,len(X.basis())):
        s=str(X.basis()[i])
        for j in range(0,len(first_term_list)):
            if first_term_list[j] not in s:
                entry=0
            else:
                upper=s.find(first_term_list[j])
                if upper==0:
                    entry=1
                else:
                    lower=s[0:upper].rfind(' ')
                    if lower==-1:
                       entry=Rational(s[0:upper-1])
                    else:
                        if s[lower+1:upper-1]=='':
                            entry=1
                            if s[lower-1]=='-':
                                entry=(-1)*entry
                        else:
                            entry=Rational(s[lower+1:upper-1])
                            if s[lower-1]=='-':
                                entry=(-1)*entry
            Transformation[i][j]=entry
    return Matrix(Transformation)
```

Solving for Linear Combinations for Annihilator Ideal Elements.

```
def vectorize(rows):
    row=[]
    for i in range(0,len(rows)):
        row.extend(rows[i])
    return row

def Form_Big_Matrix(N,S,g,Hecke_Basis):
    Transformation=Find_Basis_Transformation(N,S,g)
    rows=[vectorize((Transformation*Hecke_Basis[0]).rows())]
    for i in range(1,g):
        rows.append(vectorize((Transformation*Hecke_Basis[i]).rows()))
        print i
    return Matrix(rows)
```

Building the Relation Matrix.

```
def Clear_Denoms(matrix):
    denom=1
    for i in range(0,matrix.nrows()):
        for j in range(0,matrix.ncols()):
            if not matrix[i][j].denominator().divides(denom):
denom=denom*matrix[i][j].denominator()
    return denom*matrix

def Relation_Matrix(Hecke_Basis,kernel_matrix):
    N=kernel_matrix
    Relation_Matrix=Matrix(ZZ,N.ncols(),0)
    for i in range(0,N.nrows()):
        sum=0
        for j in range(0,N.ncols()):
            c=N[i][j]
            if c!=0:
                sum=sum+c*Hecke_Basis[j]
        Relation_Matrix=Relation_Matrix.augment(Clear_Denoms(sum).transpose())
    return Relation_Matrix
```

Finding Torsion Orders.

```
def graborder(matrix):
    order=1;
    divs=matrix.elementary_divisors();
    for i in range(0,len(divs)):
        if divs[i]>1:
            order=order*divs[i];
    return order;
```

Algorithms 1 and 2.

```
def Main(X,Y):
    N=X.ambient().cuspidal_submodule()
    time=cputime()
    g=Gamma0(N.level()).genus()
    hecks=Basis_For_TQ(N,g)
    N1=Form_Big_Matrix(N,X,g,hecks[0])
    N2=Form_Big_Matrix(N,Y,g,hecks[0])
    K1=N1.kernel().matrix()
    K2=N2.kernel().matrix()
    BottomMatrix1=Relation_Matrix(hecks[0],K1)
    BottomMatrix2=Relation_Matrix(hecks[0],K2)
    TopMatrix=BottomMatrix1.augment(BottomMatrix2)
    Numerator=graborder(TopMatrix)
    denom1=graborder(BottomMatrix1)
    denom2=graborder(BottomMatrix2)
    return Numerator/(denom2*denom1)


def Main(X,Y,p):
    N=X.ambient().cuspidal_submodule()
    g=Gamma0(N.level()).genus()
    hecks=Basis_For_TQ(N,g)
    N1=Form_Big_Matrix(N,X,g,hecks[0])
    N2=Form_Big_Matrix(N,Y,g,hecks[0])
    K1=N1.kernel().matrix()
    K2=N2.kernel().matrix()
    BottomMatrix1=Relation_Matrix(hecks[0],K1)
    BottomMatrix2=Relation_Matrix(hecks[0],K2)
    TopMatrix=BottomMatrix1.augment(BottomMatrix2)
    TopMatrix=Matrix(GF(p),TopMatrix)
    BottomMatrix1=Matrix(GF(p),BottomMatrix1)
    BottomMatrix2=Matrix(GF(p),BottomMatrix2)
    TopMatrix=Matrix(ZZ,TopMatrix)
    BottomMatrix1=Matrix(ZZ,BottomMatrix1)
    BottomMatrix2=Matrix(ZZ,BottomMatrix2)
    Numerator=graborder(TopMatrix)
    denom1=graborder(BottomMatrix1)
    denom2=graborder(BottomMatrix2)
    return Numerator/(denom2*denom1)
```

.3.2 Algorithms 3 and 4

Decomposing N and Computing Projection Maps.

```
def Find_power_generator(N):
    found=false
    i=1
    sum=0
    while not found:
        i=i+1
        t=N.T(i).matrix()
        f=t.charpoly()
        g=f.derivative()
        if GCD(f,g).is_constant():
            found=true
            t_in_ambient=N.ambient().T(i)
        sum=sum+(1/i)*N.ambient().T(i)
        if i==13:
            t_in_ambient=sum
            t=t_in_ambient.matrix().restrict(N.module())
            found=true
            f=t.charpoly()
            if not GCD(f,f.derivative()).is_constant():
               1/0
    return t,t_in_ambient



def Decompose(N,t):
    V=N.module().matrix()
    f=t.minpoly()
    factorlist=f.factor()
    factors=[]
    for i in range(0,len(factorlist)):
        factors.append(factorlist[i][0])
    structures_of_simple_spaces=[]
    for i in range(0,len(factors)):
        mat=factors[i](t).kernel().matrix()*V
        structures_of_simple_spaces.append(mat)
    exception_raiser=[]
    for structure in structures_of_simple_spaces:
        if structure.rank()>1:
            exception_raiser.append(structure)
    exception_raiser=exception_raiser[0]
    return structures_of_simple_spaces
```

```
def Get_projections_if_degree_is_bigger_than_one(N,structures):
    projections=[]
    O=N.ambient().cuspidal_submodule().old_submodule().module().matrix()
    E=N.ambient().eisenstein_subspace().module().matrix()
    other=O.stack(E).transpose()
    lengths=[structures[0].rank()]
    mat=structures[0].transpose()
    for i in range(1, len(structures)):
        lengths.append(structures[i].rank())
        mat=mat.augment(structures[i].transpose())
    mat=mat.augment(other)
    mat=mat.inverse()
    count=0
    for i in range(0,len(lengths)):
        rows=[]
        for j in range(count,count+lengths[i]):
            rows.append(mat[j])
            count=count+1
        this_proj=matrix(rows)
        if this_proj.rank()>1:
            projections.append(this_proj)
    return projections
```

Building the Matrices A_{G_f}.

```
def get_this_Galois_orbits_Hecke_basis(N,space_structure,t_in_ambient):
    rows=[]
    for row in space_structure:
        rows.append(row.list())
    t=t_in_ambient.matrix().restrict(N.module().subspace(rows))
    basis=[]
    for i in range(0,len(rows)):
        basis.append(t^i)
    return basis


def get_list_of_hecke_bases(N,structures_of_simple_spaces,t_in_ambient):
    list_of_hecke_bases=[]
    for structure in structures_of_simple_spaces:
        if structure.rank()>1:
            list_of_hecke_bases.append(get_this_Galois_orbits_Hecke_basis(N,structure,t_in_am
    return list_of_hecke_bases
```

```
def Get_symbols_with_nonzero_projections_on_each_orbit(projections,N):
    Bases_of_test_symbols=[]
    for i in range(0,len(projections)):
        Bases_of_test_symbols.append([])
    for i in range(0,len(Bases_of_test_symbols)):
        j=0
        found=false
        while not found:
            s=vector(N.ambient().basis()[j].list())
            if not (projections[i]*s).is_zero():
                Bases_of_test_symbols[i].append(j)
                found=true
            j=j+1
    return Bases_of_test_symbols



def Build_A(this_orbits_basis_of_symbols,symbols_to_needed_powers,N,proj_matrix):
    top_power=proj_matrix.nrows()
    M=N.ambient()
    columns=[]
    for i in range(0,top_power):
        columns.append([])
    for i in this_orbits_basis_of_symbols:
        for list in symbols_to_needed_powers:
            if M.basis()[i]==list[0]:
                for j in range(0,top_power):
                    y=vector(list[j].list())
                    proj=(proj_matrix*y).list()
                    columns[j].extend(proj)
    return matrix(QQ,columns).transpose()



def get_symbols_to_needed_powers(t_in_ambient,N,Bases_of_test_symbols,sizes):
    M=N.ambient()
    all_indices_of_importance=[]
    for b in Bases_of_test_symbols:
        all_indices_of_importance.extend(b)
    all_indices_of_importance=list(set(all_indices_of_importance))
    symbols_to_needed_powers=[]
    for i in all_indices_of_importance:
        this_symbols_images_under_powers=[]
        max_power_for_this_symbol=0
        for j in range(0,len(Bases_of_test_symbols)):
            if (i in Bases_of_test_symbols[j] and (sizes[j]-1)>max_power_for_this_symbol):
                max_power_for_this_symbol=sizes[j]-1
        this_symbols_images_under_powers.append(M.basis()[i])
        for j in range(0,max_power_for_this_symbol):
            this_symbols_images_under_powers.append(t_in_ambient(this_symbols_images_under_po
        symbols_to_needed_powers.append(this_symbols_images_under_powers)
    return symbols_to_needed_powers

def Build_Af_for_each_orbit(t_in_ambient,Bases_of_test_symbols,N,projections):
    sizes=[]
    for mat in projections:
        sizes.append(mat.nrows())
    symbols_to_needed_powers=get_symbols_to_needed_powers(t_in_ambient,N,Bases_of_test_symbol
    Af_matrices=[]
    for i in range(0,len(Bases_of_test_symbols)):
        Af_matrices.append(Build_A(Bases_of_test_symbols[i],symbols_to_needed_powers,N,projec
    return Af_matrices
```

Computing b_{G_f}.

```
def Build_bf_for_each_orbit(Bases_of_test_symbols,N,projections,p):
    all_indices_of_importance=[]
    bfs=[]
    M=N.ambient()
    for i in range(0,len(Bases_of_test_symbols)):
        bfs.append([])
    for b in Bases_of_test_symbols:
        all_indices_of_importance.extend(b)
    all_indices_of_importance=list(set(all_indices_of_importance))
    for i in all_indices_of_importance:
        y=M._compute_hecke_matrix_prime(p,rows=[i]).transpose()
        for j in range(0,len(Bases_of_test_symbols)):
            if i in Bases_of_test_symbols[j]:
                bfs[j].extend(projections[j]*y)
    for i in range(0,len(bfs)):
        change_back_to_vector=[]
        for j in range(0,len(bfs[i])):
            change_back_to_vector.append(bfs[i][j][0])
        bfs[i]=vector(change_back_to_vector)
    return bfs
```

Computing the Jordan Cannonical Form of $T_{p|\mathbf{N}}$.

```
def get_matrix_for_T_acting_on_only_this_Galois_orbit(basis,A,b):
    c=A.solve_right(b)
    sum=0
    for i in range(0,len(c)):
        sum=sum+c[i]*basis[i]
    return sum

def get_list_of_Tp_on_all_Galois_orbits(list_of_hecke_bases,Af_matrices,bfs):
    list_of_Tp_on_orbits=[]
    for i in range(0,len(bfs)):
        list_of_Tp_on_orbits.append(get_matrix_for_T_acting_on_only_this_Galois_orbit(list_of
    return list_of_Tp_on_orbits
```

Filling in the Array T.

```
def Get_primes(N,prec):
    Tns=[]
    for i in range(0,prec):
        Tns.append([])
    d=N.character()
    time=cputime()
    pair=Find_power_generator(N)
    print('power generator' + ' ' + str(cputime()-time))
    t=pair[0]
    index=pair[1]
    time=cputime()
    structs=Decompose(N,t)
    print('decomposition' + ' ' + str(cputime()-time))
    time=cputime()
    projs=Get_projections_if_degree_is_bigger_than_one(N,structs)
    print('projections' + ' ' + str(cputime()-time))
    time=cputime()
    hecks=get_list_of_hecke_bases(N,structs,index)
    print('hecks'+' ' + str(cputime()-time))
    time=cputime()
    symbas=Get_symbols_with_nonzero_projections_on_each_orbit(projs,N)
    print('symbas'+ ' ' + str(cputime()-time))
    time=cputime()
    Af_matrices=Build_Af_for_each_orbit(index,symbas,N,projs)
    print('Af_matrices'+ ' ' + str(cputime()-time))
    time=cputime()
    for p in prime_range(2,prec+1):
        print(p)
        bfs=Build_bf_for_each_orbit(symbas,N,projs,p)
        Tps=get_list_of_Tp_on_all_Galois_orbits(hecks,Af_matrices,bfs)
        Tns[p]=Tps
    print('ran loop' +' ' + str(cputime()-time))
    return [Tns,d]

def get_prime_powers(Tns,d):
    B=len(Tns)
    orbits=len(Tns[3])
    for p in prime_range(1,floor(sqrt(B))+1):
        i=2
        while p^i<B:
            if i==2:
                for j in range(0,orbits):
                    Tns[p^i].append(Tns[p][j]*Tns[p^(i-1)][j]-d(p)*p*identity_matrix(QQ,Tns[p
            else:
                for j in range(0,orbits):
                    Tns[p^i].append(Tns[p][j]*Tns[p^(i-1)][j]-d(p)*p*Tns[p^(i-2)][j])
            i=i+1
    return Tns




def hecke_general_product(Tns):
    i=2
    B=len(Tns)
    orbits=len(Tns[3])
    while i<B:
        if not i.is_prime_power():
            for j in range(0,orbits):
                prod=identity_matrix(QQ,Tns[3][j].ncols())
                for prime,power in factor(i):
                    prod=prod*Tns[prime^power][j]
                Tns[i].append(prod)
        i=i+1
    return(Tns)
```

Including Degree One Subspaces.

```
def Include_degree_one_spaces(N,bound):
    conductor=N.level()
    basis_for_integer_newforms=[]
    for E in cremona_optimal_curves(srange(conductor,conductor+1)):
        basis_for_integer_newforms.append(E.newform().coefficients(bound-1))
    return matrix(basis_for_integer_newforms)
```

Algorithm 3.

```
def Get_new_space(level,prec):
    N=ModularSymbols(level,sign=1).cuspidal_subspace().new_subspace()
    curves=Include_degree_one_spaces(N,prec)
    try:
        tns=Get_primes(N,prec)
    except:
        return curves
    tns=get_prime_powers(tns[0],tns[1])
    tns=hecke_general_product(tns)
    for j in range(0,len(tns[2])):
        tns[1].append(identity_matrix(QQ,tns[2][j].ncols()))
    cols=[]
    for i in range(1,prec):
        this_column=[]
        for tn in tns[i]:
            this_column.extend(tn[0].list())
        cols.append(this_column)
    q_expansion_matrix=matrix(cols).transpose()
    if curves.rank()>0:
        q_expansion_matrix=q_expansion_matrix.stack(curves)
    return q_expansion_matrix
```

Degeneracy Maps.

```
def degenerate(row,div):
    image=[]
    for i in range(0,len(row)):
        if div.divides(i+1):
            r=int((i+1)/div)-1
            image.append(row[r])
        else:
            image.append(0)
    return image
```

Algorithm 4.

```
def Get_space_at(N,B):
    space=[]
    for d in divisors(N):
        lower_level=int(N/d)
        rows=[]
        mat=Get_new_space(lower_level,B)
        for row in mat:
            rows.append(row.list())
        for row in rows:
            for div in d.divisors():
                space.append(degenerate(row,div))
    return space
```

Algorithm 5.

```
def Lift_to_Z(B):
    Bs=[]
    for i in range(0,len(B)):
        lift_big_entries=vector(ZZ,vector(GF(p),Ech_Form[i]))
        lift_small_entries=[]
        for j in range(0,len(lift_big_entries)):
            if lift_big_entries[j]<=p/2:
                lift_small_entries.append(reduced_vector_without_abs_value_condition[j])
            else:
                lift_small_entries.append(-(p-reduced_vector_without_abs_value_condition[j]))
        Bs.append(lift_small_entries)
    return Bs


def p_saturation(L,p):
    print('starting')
    n=len(L)
    N=len(L[0])
    if N<n:
        raise IndexError, "vectors contain too few entries"
    Us=[]
    for i in range(0,n):
        this_row=[]
        for j in range(0,N):
            this_row.append(L[i][j] % p)
        Us.append(vector(this_row))
    B=matrix(GF(p),Us).kernel().basis()
    if len(B)==0:
        return L
    Ech_Form=matrix(GF(p),B).echelon_form()
    Bs=Lift_to_Z(B)
    Vs=[]
    for i in range(0,n):
        Vs.append(L[i])
    for i in range(0,len(Bs)):
        ##find leading 1 position and call it l
        l=0
        while Bs[i][l]==0:
            l=l+1
        p_times_W=0
        for j in range(0,n):
            p_times_W=p_times_W+Bs[i][j]*vector(ZZ,L[j])
        W=p_times_W/p
        try:
            Vs[l]=vector(ZZ,W)
        except TypeError:
            print('W not coercible to ZZ')
    return p_saturation(Vs,p)
```

Algorithm 6.

```
def sat(M):
    gcd=0
    count=0
    exception_raiser=0
    N=M.ncols()
    n=M.nrows()
    import sage.matrix.matrix_integer_dense_saturation as s
    while count<3:
        cols=[]
        for i in range(0,n):
            sum=0
            for col in s.random_sublist_of_size(N,10):
                sum=sum+randint(-10,10)*M.column(col)
            cols.append(sum)
        det=matrix(cols).determinant()
        if det>0:
            gcd=GCD(gcd,det)
            count=count+1
        exception_raiser=exception_raiser+1
        if exception_raiser>100:
            1/0
    rows=M.rows()
    for p in gcd.prime_divisors():
        print('starting saturaton at' + ' ' + str(p))
        rows=p_saturation(rows,p)
    return matrix(rows)
```

Algorithm 7.

```
def intbas(N,B):
    mat=matrix(Get_space_at(N,B))._clear_denom()[0]
    return sat(mat)
```

.3.3 Algorithm 8

Checking if a Prime is a Congruence Prime.

```
def is_congruent(E,rows,p,sb):
    b=false
    E=E.newform().coefficients(sb)
    q=Matrix(GF(p),rows).row_space().dimension()
    rows.append(E)
    if Matrix(GF(p),rows).row_space().dimension()<q+1:
        b=true
    rows.pop()
    return b
```

Listing All Congruence Primes.

```
def congruence_primes(E,rows,sb):
    con_primes=[]
    potentials=E.modular_degree().prime_divisors()
    level=E.conductor()
    if not is_squarefree(level):
        pds=level.prime_divisors()
        for i in range(0,len(pds)):
            if pds[i].divides(level/pds[i]):
                if not pds[i] in potentials:
                    potentials.append(pds[i])
    for i in range(0,len(potentials)):
        if is_congruent(E,rows,potentials[i],sb):
            con_primes.append(potentials[i])
    if con_primes==[]:
        con_primes.append(1)
    return con_primes
```

REFERENCES

[1] Amod Agashe. Unpublished manuscript.

[2] Amod Agashe. A visible factor of the special L-value. *J. Reine Angew. Math.*, 644:159–187, 2010.

[3] Amod Agashe, Kenneth Ribet, and William A. Stein. The Manin constant. *Pure Appl. Math. Q.*, 2(2, part 2):617–636, 2006.

[4] Amod Agashe and William Stein. Visible evidence for the Birch and Swinnerton-Dyer conjecture for modular abelian varieties of analytic rank zero. *Math. Comp.*, 74(249):455–484, 2005. With an appendix by J. Cremona and B. Mazur.

[5] A. O. L. Atkin and J. Lehner. Hecke operators on $\Gamma_0(m)$. *Math. Ann.*, 185:134–160, 1970.

[6] Henri Darmon, Fred Diamond, and Richard Taylor. Fermat's last theorem. In *Current developments in mathematics, 1995 (Cambridge, MA)*, pages 1–154. Int. Press, Cambridge, MA, 1994.

[7] Fred Diamond and Jerry Shurman. *A first course in modular forms*, volume 228 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2005.

[8] Matthew Emerton. Optimal quotients of modular Jacobians. *Math. Ann.*, 327(3):429–458, 2003.

[9] Benedict H. Gross and Don B. Zagier. Heegner points and derivatives of L-series. *Invent. Math.*, 84(2):225–320, 1986.

[10] Haruzo Hida. Congruence of cusp forms and special values of their zeta functions. *Invent. Math.*, 63(2):225–261, 1981.

[11] Dino Lorenzini. Torsion and tamagawa numbers (preprint).

[12] B. Mazur. Modular curves and the Eisenstein ideal. *Inst. Hautes Études Sci. Publ. Math.*, (47):33–186 (1978), 1977.

[13] Loïc Merel. Universal Fourier expansions of modular forms. In *On Artin's conjecture for odd 2-dimensional representations*, volume 1585 of *Lecture Notes in Math.*, pages 59–94. Springer, Berlin, 1994.

[14] Kenneth A. Ribet. Mod p Hecke operators and congruences between modular forms. *Invent. Math.*, 71(1):193–205, 1983.

[15] K. A. Ribet. On modular representations of $\mathrm{Gal}(\overline{\mathbf{Q}}/\mathbf{Q})$ arising from modular forms. *Invent. Math.*, 100(2):431–476, 1990.

[16] René Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7(1):219–254, 1995. Les Dix-huitièmes Journées Arithmétiques (Bordeaux, 1993).

[17] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1986.

[18] William Stein. *Modular forms, a computational approach*, volume 79 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2007. With an appendix by Paul E. Gunnells.

[19] W. A. Stein et al. *Sage Mathematics Software (Version 4.6.2)*. The Sage Development Team, 2011. `http://www.sagemath.org`.

[20] J. Sturm. On the congruence of modular forms. In *Number theory (New York, 1984–1985)*, pages 275–280. Springer, Berlin, 1987.

[21] Mark Watkins. Computing the modular degree of an elliptic curve. *Experiment. Math.*, 11(4):487–502 (2003), 2002.

[22] Andrew Wiles. Modular elliptic curves and Fermat's last theorem. *Ann. of Math. (2)*, 141(3):443–551, 1995.

BIOGRAPHICAL SKETCH

Randy Heaton grew up in Perry, Georgia and later Panama City Beach, Florida. He graduated from Georgia Tech and began graduate school at Florida State University in 2006.