2015

Random Matrix Theory and the Attraction of Zeros of L-Functions From the Central Point

Katherine Elizabeth Harris
keh021@bucknell.edu

RANDOM MATRIX THEORY AND THE ATTRACTION OF ZEROS OF L-FUNCTIONS FROM THE CENTRAL POINT

by

Katherine Harris

A Thesis

Presented to the Faculty of

Bucknell University

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science with Honors in Mathematics

April 24, 2015

Approved:

Nathan Ryan
Thesis Advisor

Thomas Cassidy
Chair, Department of Mathematics

Acknowledgments

Thanks to all of the faculty in the Math Department here at Bucknell that have made this experience possible to me, as well as faculty from other departments who have encouraged me in my mathematical pursuits. Thanks to my parents and sister, who have no idea what I study, but are supportive no matter what. Finally, a special thanks to my thesis advisor, Nathan Ryan. He was not only instrumental in his position as advisor for my thesis, but also undescribably helpful in my graduate school search and many of the other trials and tribulations of my last year of undergraduate study.

Contents

List of Figures

Abstract

We explore the attraction of zeros near the central point of L-functions associated with elliptic curves and modular forms. Specifically, we consider families of twists of elliptic curves, the family of weight 2 modular forms, and the family of level 1 modular forms. We observe experimentally an attraction of the zeros near the central point, and that the attraction decreases with the rank r of the L-function. However, for each set of L-functions of rank r within a particular family we observe a statistically significant increase in the attraction as the conductors of the L-functions increase. This indicates a correspondence with the random matrix theory result about the vanishing of the distance between eigenangles near 1 as the size of the matrix increases, but also that this correspondence only exists in the limit, since we observe less attraction otherwise. Additionally, we begin preliminary investigation on a new statistic, the relationship between the value of the first zero above the central point and the value of the L-function at $s = \frac{1}{2}$.

Chapter 1

Introduction

Many of the most important problems considered by number theorists over the years are problems involving prime numbers. Prime numbers have been studied by mathematicians for thousands of years. During this time, many important results involving primes have been proven. For example, it has been proven that there are an infinite number of primes. We also know results concerning the distribution of the primes over the integers. One important result is the Prime Number Theorem [16]:

Theorem 1. *The number of primes $p \leq x$,$\pi(x)$, satisfies the formula*

$$\pi(x) \approx \frac{x}{\log(x)}$$

as $x \to \infty$.

This theorem states that the distribution of the primes is asymptotic in the limit to $\frac{x}{\log(x)}$. This concept is illustrated in Figure 1.1 [4].

However, we note from the figure that there is a gap between this asymptotic behavior, and the actual true distribution of the of the primes. For years, mathematicians searched for ways to close this gap. In the end, it was Bernard Riemann who discovered another possible way of looking at the distribution of primes. He did this by considering the Riemann ζ-function, written as

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

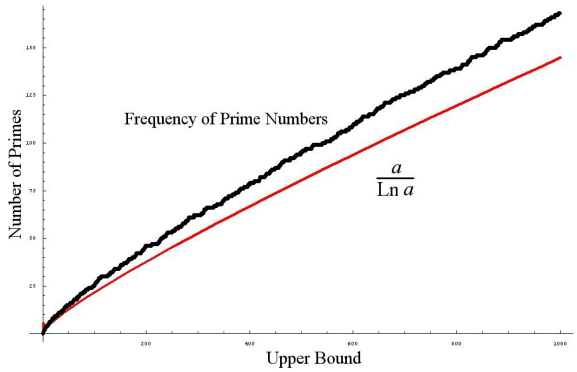


Figure 1.1: Prime Number Theorem

The ζ-function is a function on the complex plane that is defined for $\text{Re}(s) > 1$. Interestingly, the ζ-function has a connection to the primes known as the *Euler product* [17].

Theorem 2. *For any $s \in \mathbb{C}$, the Riemann ζ-function has a product expansion*

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}}$$

where both sides of the equation converge for $Re(s) > 1$.

We note that the Euler product implies that none of the zeros of the Riemann *zeta*-function occur for $\text{Re}(s) > 1$. It is also true that that Riemann ζ-function extends analytically onto the entire complex plane, except a simple pole at $s = 1$. Let

$$\Lambda(s) := \pi^{-s/2}\Gamma\left(\frac{s}{2}\right)\zeta(s).$$

Then

$$\Lambda(s) = \Lambda(1 - s).$$

We refer to this as the *functional equation* of the ζ-function [17].

The Riemann Hypothesis, proposed by Bernhard Reimann in 1859, is stated as follows: *All non-trivial zeros of the Riemann zeta function have real part $s = \frac{1}{2}$* [17]. Looking back, the Prime Number Theorem is equivalent to saying that there are no zeros of the ζ-function on the line $s = 1$. The Riemann Hypothesis, however, implies a much stronger result than the Prime Number Theorem. In fact, it tells us exactly

that all of the zeros of the ζ-function lie on the line $\mathrm{Re}(s) = \frac{1}{2}$, or that the zero-free region is as large as possible.

The Riemann ζ-function is an example of an L-function. Just like the ζ-function, all L-functions are defined by a Dirichlet series, an Euler product, a functional equation, and an analytic continuation. This allows us to use the generalized Riemann Hypothesis, which states that all of the zeros for any L-function lie on the line $\mathrm{Re}(s) = \frac{1}{2}$. L-functions also have a property N, level, which can be used as a partial ordering on collections of L-functions. Finally, the *rank* of an L-function is the order of $L(1/2)$, where the order of $L(1/2)$ is $r + 1$ such that $L^r(1/2) = 0$ but $L^i(1/2) \neq 0$ for all $0 \leq i < r$.

In addition to quesitons about primes, one of the problems that number theorists often consider is finding solutions to polynomials over the rationals. While linear and quadratic polynomial solutions are often covered in an undergraduate number theory course, the problem becomes significantly more complicated when we consider equations $y^2 = f(x)$ where $f(x)$ is a cubic $f \in \mathbb{Q}[x]$. In this case, number theorists introduce objects known as elliptic curves over $\mathbb{Q}$. To find solutions of elliptic curves over the rationals, we consider solutions over $\mathbb{F}_p$, the finite field modulo p. For $f(x)$ a cubic polynomial, we can define

$$E(\mathbb{F}_p) = \{(x_0, y_0) \in \mathbb{F}_p^2 : y_0^2 = f(x_0)\}.$$

We have

$$a_p = p + 1 - |E(\mathbb{F}_p)|$$

. We define the Euler product, and with it the L-function associated with E as the Dirichlet series

$$L(E, s) = \sum_{n=1}^{\infty} \frac{a_n}{n^s} = \prod_{p \text{ prime}} \frac{1}{1 - a_p p^{-s} + p^{1-2s}}.$$

In this way, we are always able to find a unique L-function associated with an elliptic curve. A theorem from Mordell-Weil tells us an important result about elliptic curves [10].

Theorem 3. *The solutions to an elliptic curve over$\mathbb{Q}$ form a finitely generated abelian group.*

As we will show in Chapter 2, it is self-evident that the solutions to an elliptic curve form an abelian group by exploration of the group laws. What is important about this theorem, however, is that the group is finitely generated. Finitely generated groups

are guaranteed to have finite *rank*. This leads us to the Birch and Swinnerton-Dyer Conjecture (1965), which is stated as: *the rank of the abelian group* $E(\mathbb{Q})$ *is equal to the rank of the associated L-function* $L(E, s)$ [24].

These results tell us that we can compare families of L-functions of a particular rank to abelian groups a certain rank. Thus we have found a number theoretic reason for studying families of L-functions.

Modular forms are a type of infinite polynomial sum whose coefficients are of particular interest to number theorists due to their unique algebraic and analytic properties. Modular forms are defined by two main parameters: weight N and level k.

A famous theorem, widely known as the Modularity Theorem, due to Wiles [23], Taylor-Wiles [21], Breuil-Conrad-Diamond-Taylor [7], says:

Theorem 4. *Let* E *be an elliptic curve over* $\mathbb{Q}$ *of conductor* N*. Then there exists a modular form* F *of weight* 2 *and level* N *such that* $L(F, s) = L(E, s)$*.*

This tells us that elliptic curves make up a subset of modular forms of weight 2 and level N. In particular, we note that this is a very small subset of all modular forms, since modular forms exists for many weights k and levels N. Therefore, we are not only interested in finding L-functions associated elliptic curves, but also those L-functions that are associated with modular forms as well.

There exist interesting conjectures as to the attraction of zeros of L-functions to the central point $L(\frac{1}{2})$ on the complex plain. These conjectures result from a heuristic correspondance between L-functions and a field known as random matrix theory. In April 1972, Hugh Montgomery was visiting Princeton University to share a result involving the statistics of the Riemann ζ-function, defined above. During his visit, he happened to discuss his result with Freeman Dyson, a physicist working at the university, over tea. Through the course of their discussion, the two discovered a remarkable correspondence between Montgomery's work and the eigenvalues being studied in random matrix theory prmtback. Ever since this discovery, the understanding of the connection between the zeros of L-functions and random matrix theory has been an area of interest for many mathematicians.

In particular, the correspondence that we are interested in has to do with the attractions that occurs for the smallest zeros found on the critical line. In random matrix theory, a phenomenon has been observed where attraction of the smallest

eigenvalues to 0 increases to 0 as the size of the matrix tends towards infinity. It has been observed that trends having to do with the size of a random matrix correspond to trends involving the conductor size of an L-function, a parameter which is determined by certain parameters of its associated number theoretic object. Therefore, we expect that as we increase the conductor of L-functions towards infinity, the attraction between the first zero on the critical line and the central point should be maximized.

The first experiment of this kind was done by Steven J. Miller [14] at Brown University almost a decade ago. In his experiment, Miller formulates the conjecture that the zeros of L-functions with the parameters weight 2 and level N are attracted to each other more as the conductor (which in the case of weight 2 is equal to the level) increases. Specifically, he ran experiments for certain families of L-functions determined by elliptic curves, and in all cases observed that that for finite N, the attraction is less than the maximal limit case. As conductor size N increases, the behavior of the attraction tends towards the maximal random matrix theory prediction, but does not reach this limit in the finite case.

In our experiments, we verify this correspondence for larger families of L-functions associated with elliptic curves. We also verify the connection for different families of L-functions associated with modular forms up to a certain conductor. Finally, we compute and observe a new statistic linking the value of an L-function at the central point to its first zero on the critical line.

The connection between random matrix theory and the functions involved in my experiment is fortunate as it places a broader relevance and significance on the results and analysis of the data. Currently, Miller is working at Williams with Harvard undergraduate student Patrick Ryan on developing and proving the abstract random matrix theory results corresponding to the experiment we conducted. Therefore, while the experimental results of our research are of immediate interest to mathematicians since they have never been done before, the broader analysis and comparison of the experimental results to the abstract results being studied by Miller and Ryan could have an even greater level of interest amongst broader fields within the mathematical community.

Chapter 2

Modular Forms

2.1 Defining Modular Forms

We will begin our discussion about modular forms with some standard definitions and results about modular forms, which can be found in Diamond and Shurman's graduate textbook on the subject [9]. We start by considering the group $SL_2(\mathbb{Z})$:

$$SL_2(\mathbb{Z}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} : a, b, c, d \in \mathbb{Z}, ad - bc = 1 \right\}.$$

We call $SL_2(\mathbb{Z})$ the *modular group*, and from now on we will represent it using the symbol Γ. For our purposes, it is important to realize that within the group Γ, we have the following subgroups:

Definition 1. *For $N \in \mathbb{N}$, the congruence subgroup of level N is defined by*

$$\Gamma_0(N) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma : c \equiv 0 \ mod\ N \right\}.$$

We pause to verify that $\Gamma_0(N)$ is a subgroup of Γ.

Proposition 1. *$\Gamma_0(N)$ is a subgroup of Γ*

Proof: By the properties of matrices, it is clear the that the property of associativity holds. We first check that $\Gamma_0(N)$ is closed under the group operation. We

compute

$$\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} = \begin{pmatrix} a_1a_2 + b_1c_2 & a_1b_2 + b_1d_2 \\ c_1a_2 + d_1c_2 & c_1b_2 + d_1d_2 \end{pmatrix}$$

where $c_1, c_2 \equiv 0 \mod N$. Then $c_1a_2 + d_1c_2 \equiv 0 \mod N$, and thus the subgroup is closed under matrix multiplication. Also, the identity matrix I is included in the subgroup, since in that case $c = 0$, which is obviously equivalent to $0 \mod N$. Finally, we consider inverses. We know that the inverse of the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is $\frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$. However, since we are in Γ, $ad - bc = 1$. Thus we must only check $-c \equiv 0 \mod N$. Since $c \equiv 0 \mod N$, this statement holds, and thus $\Gamma_0(N)$ is a subgroup of Γ. $\square$

In addition to $\Gamma_0(N)$ being a subgroup, we also must note that there exists a group action on the upper half of the complex plane $\mathbb{H} = \{z \in \mathbb{C} : \text{Im } z > 0\}$, defined as

$$\gamma z \mapsto \frac{az + b}{cz + d}$$

for $\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_0(N)$.

Proposition 2. *The operation described above is a group action.*

Proof: We first consider the identity element $\gamma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Then

$$\gamma z \to \frac{1z + 0}{0z + 1} = z,$$

and thus is the identity as required. We now consider associativity. Consider γ_1 and γ_2 such that

$$\gamma_i = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}.$$

Then we compute

$$\begin{aligned}
(\gamma_1\gamma_2)(z) &= \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} (z) \\
&= \begin{pmatrix} a_1a_2 + b_1c_2 & a_1b_2 + b_1d_2 \\ c_1a_2 + d_1c_2 & c_1b_2 + d_1d_2 \end{pmatrix} (z) \\
&= \frac{(a_1a_2 + b_1c_2)z + a_1b_2 + b_1d_2}{(c_1a_2 + d_1c_2)z + c_1b_2 + d_1d_2} \\
&= \frac{a_1\frac{a_2z+b_2}{c_2z+d_2} + b_1}{c_1\frac{a_2z+b_2}{c_2z+d_2} + d_2} \\
&= \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} \frac{a_2z + b_2}{c_2z + d_2} \\
&= \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} \left(\begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} (z) \right) \\
&= \gamma_1(\gamma_2 z)
\end{aligned}$$

Thus associativity holds. Finally, we need to show that if $z \in \mathbb{H}$, then $\gamma z \in \mathbb{H}$. We let $z = x + iy$. Then we note that

$$\frac{a(x+iy)+b}{c(x+iy)+d} = \frac{(ax+b)+aiy}{(cx+d)+ciy}.$$

To isolate the imaginary part of this fraction, we compute

$$\frac{(ax+b)+aiy}{(cx+d)+ciy} \cdot \frac{(cx+d)-ciy}{(cx+d)-ciy} = \frac{((ax+b)(cx+d)-ay^2)+((ax+b)(-ciy)+(cx+d)(aiy))}{(ax+d)(cx+d)+c^2y^2}.$$

Then we must only verify that $(ax+b)(-c)+(cx+d)(a) = ad-bc$. But we recall from our definition of $\Gamma_0(N)$ that $a, b, c, d \in \mathbb{Z}$, with $ad - bc = 1$. Thus $\gamma z \in \mathbb{H}$ and the group action is valid. $\square$

We now have the machinary necessary in order to define a modular form. A *modular form* is an analytic function on the upper half of the complex plane. It enjoys symmetries induced by the group action on the modular group Γ, and has a Fourier series of a particular form. It is partially specified by two parameters: the level N and the weight k. These two characteristics will be helpful for us later on for performing computations. We formalize the specifics of this definition using language found in [18] as follows:

Definition 2. *Let $k, N \in \mathbb{N}$. An analytic function $f : \mathbb{H} \to \mathbb{C}$ in the upper half of the complex plane is called a modular form of weight k with respect to $\Gamma_0(N)$ if for all $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_0(N)$,*

$$f\left(\frac{az+b}{cz+d}\right) = (cz+d)^k f(z),$$

and

$$f(z) = \sum_{n=0}^{\infty} a_n q^n$$

for $q = e^{2\pi i n z}$. We call the numbers a_n the Fourier coefficients of f, and say that in this case the Fourier expansion of f has finite principal part.

We note that since $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \in \Gamma_0(N)$, $f(z+1) = f(z)$, and so the Fourier expansion of f exists.

From this definition, we will prove an important fact about the weight k of a modular form.

Proposition 3. *The weight of a modular form k is always even.*

Proof: Suppose instead that the weight k of a modular form f is odd, and consider $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \in \Gamma_0(N)$. Then we compute

$$f\left(\frac{az+b}{cz+d}\right) = f\left(\frac{-z}{-1}\right) = f(z) \neq -f(z) = (-1)^k f(z) = (cz+d)^k f(z).$$

Thus, we have created a contradiction in the definition, and our proposition must be true. □

We also note that in some cases we will wish to consider a slightly different formulation of the group action in the above definition. Recall that a *meromorphic* function is simply an analytic function with an isolated set of poles.

Definition 3. *Let $f : \mathbb{H} \to \mathbb{C}$ be meromorphic and $k \in \mathbb{N}$. For $\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in GL_2(\mathbb{R})$, let*

$$f|_k\gamma = (\det\gamma)^{k/2}(cz+d)^{-k} f(\gamma z).$$

In Definition 3, we see that for $\gamma, \delta \in GL_2(\mathbb{R})$, f has the equivalence $(f|_k\gamma)|_k\delta = f|_k(\gamma\delta)$. Then $\circ|_k\gamma$ is a linear operator on the group. Now recall the action on the modular group that was included as part of Definition 2. Since $\Gamma_0(N) \subset GL_2(\mathbb{R})$, we can apply this new definition to the group action, and see that $f|_k\gamma = f$ for all $\gamma \in \Gamma_0(N)$. We typically refer to this property of the action on modular forms by saying they are *invariant* under the modular group. We also note that in every case $k > 0$ and k is even.

One of the key properties of modular forms is that they have a finite dimensional vector space structure. We denote the vector space of modular forms of weight k with respect to $\Gamma_0(N)$ as $M_k(N)$. Also, there is an important subset of $S_k(N)$ known as the *cusp forms* of a particular weight k.

Definition 4. *A modular form with a zero constant coefficient in the Fourier expansion is called a cusp form.*

Let $\mathbb{H}^* = \mathbb{H} \cup \mathbb{Q} \cup \{\infty\}$. Then we say $\mathbb{H}^*$ is the set of cusps. Intuitively, the modular forms defined above are called cusp forms since they vanish at the cusps and at the points at infinity. We denote the vector space of cusp forms of weight k with respect to $\Gamma_0(N)$ as $S_k(N)$. We note that $S_k(N)$ is a subset of $M_k(N)$. Also, it can be shown that the dimensions of both of these spaces are finite.

2.2 Atkin-Lehner Eigenvalues

In designing our computations, we use certain properties of modular forms in order to organize the computations to cover the entire vector space. One property that is necessary for our computations is the Atkin-Lehner eigenvalue of a modular form.

We start by considering the vector space $M_k(N)$. We define the *Atkin-Lehner involution* to be the matrix

$$W_N = \begin{pmatrix} 0 & -1 \\ N & 0 \end{pmatrix}.$$

We note that W_N takes the name *involution* since $W_N^2 z = z$, and that $W_N \notin \Gamma_0(N)$, since $\det(W_n) \neq 1$ for any $N \in \mathbb{Z}^+, N \neq 1$.

We now take some $L = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_0(N)$. We compute

$$W_N L W_N = \begin{pmatrix} -Nd & c \\ N^2 b & -a \end{pmatrix}.$$

Since $c \equiv 0 \pmod{N}$ and $N^2 b \equiv 0 \pmod{N}$, we have

$$W_N L W_N \equiv \begin{pmatrix} -d & c/N \\ Nb & -a \end{pmatrix} \in \Gamma_0(N).$$

Recall the space $M_k(N)$ of modular forms of weight k and level N. Suppose $f \in M_k(N)$. Since $W_N L W_N \in \Gamma_0(N), f|_k(W_N L W_N) = f$, and since $W_N^2 z = z$, $f|_k W_N^2 = f$, and thus it is an involution. Then we have

$$(f|_k W_N)|_k L = f|_k W_N.$$

So $\circ|_k W_N$ is a linear operator which is an endomorphism on $M_k(N)$, i.e. a homomorphism from $M_k(N)$ to itself. Yet we recall that we know $\circ|_k W_N$ is also an involution. Thus, since all matrices that are involutions have eigenvalues ± 1, the only eigenvalues of the linear operator on $M_k(N)$ are ± 1. This conclusion leads us to the following important theorem which will allow us to determine the Atkin-Lehner eigenvalues of modular forms.

Theorem 5. *The space $M_k(N)$ is the direct sum of the two eigenspaces $M_k^+(N)$and $M_k^-(N)$ with respect to the linear operator $\circ|_k W_N$ and the eigenvalues ± 1. Let $f \in M_k^{\pm}(N)$. Then f satisifies*

$$f\left(\tfrac{-1}{Nz}\right) = \mu N^{k/2} z^k f(z),$$

where $\mu = \pm 1$ is called the Atkin-Lehner eigenvalue of f.

Note that this theorem also holds for $S_k(N)$, the space of cusp forms, since $S_k(N) \subset M_k(N)$.

2.3 Newforms

In anticipation of wishing to maximize the efficiency of our computations in our experiments, we will now turn to a type of modular forms known as a *newform*. To do this, we consider two spaces of cusp forms $S_k(N)$ and $S_k(M)$ with $N, M \in \mathbb{Z}$

such that $M|N$. By the definition of $\Gamma_0(N)$, $M|N$ implies $\Gamma_0(M) \subseteq \Gamma_0(N)$. Thus in essence, the level N of a modular group provides us with a nested set of subgroups with N ordered by divisibility. Similarly, $S_k(M) \subseteq S_k(N)$. Let $aM|N$ and $f \in S_k(M)$. Then by this containment relationship, the function $z \to f(az)$ is a modular form in $S_k(N)$.

By repeating the above process, we can find a subspace of $S_k(N)$ comprised of elements that can be found by mappings from cusp forms whose levels are proper divisors of N. We call this subspace the space of *oldforms* of $S_k(N)$. This name is logical, since if we are considering the spaces of cusp forms for a fixed k and an increasing N, the space of oldforms is that of the spaces which we have already seen in earlier spaces where $M|N$.

Oldforms are important for us since we do not wish to double count any of the same modular forms in our sets of data. Therefore, we wish to only consider the subspace of newforms of each $S_k(N)$. The subspace of newforms is the complement of the subspace spanned by the oldforms for a particular $S_k(N)$, i.e. the orthogonal complement with respect to the *Petersson inner product*:

Definition 5. *Let $f, g \in S_k(N)$. Then we define the Petersson inner product as*

$$\langle f, g \rangle = \int f(z)\overline{g(z)}y^k \frac{dxdy}{y^2}.$$

It has been verifed that the Petersson inner product is in fact a well-defined inner product for elements of $S_k(N)$ [18]. Thus we have a way of taking the orthogonal complement in $S_k(N)$.

2.4 Hecke Operators

We now explore whether there is an easier way for us to find the newforms of $S_k(N)$ besides finding all the oldforms and taking the complement of that space. In fact, there exists an object called a *Hecke operator* which provides us with a computationally efficient way to determine a basis of newforms of $S_k(N)$, defined as follows:

Definition 6. *Let $f \in S_k(N)$. Then for $n \in \mathbb{N}$, the Hecke operator $T(n)$ is such that*

$$T(n)_f = \frac{1}{n} \sum_{ad=n} a^k \sum_{0 \le b < d} f\left(\frac{az+b}{d}\right).$$

While the theory of Hecke operators covers a wide range of results and applications, for our computational purposes we require only the following theorem.

Theorem 6. *Let $T(n)$ be the set of all functions f such that $T(n)_f = \lambda_f(n)f$ for all $n \in \mathbb{N}$. The basis elements of $S_k(N)$ that are simultaneous eigenfunctions of $T(n)$ are the Hecke eigenforms. Each member f of this basis can be normalized to have the Fourier coefficient $a_1 = 1$, so $a_n = \lambda_f(n)$ for all $n \in \mathbb{N}$. Also, the coefficients a_n are multiplicative.*

As we can see, Hecke operators provide us with a concrete process which allows us to find the subspace of newforms of a particular space of cusp forms without considering the subspace of oldforms. We now have a way to find all of the cusp forms for a certain $S_k(N)$ that are not already contained in a space of cusp forms with a lower level. Thus we have devised a way for determining all of the cusp forms (without overcounting) for a fixed k while letting N range from 1 to some $N \in \mathbb{Z}$. Hence we have figured out the most efficient way of spanning the set of all computable modular forms.

2.5 Field of Definition

At this point, there exists one final object associated with a modular form that we have yet to define known as the *field of definition*. Intuitively, the field of definition of a modular form f is the field $\mathbb{F}$ over which the coefficients of the Fourier expansion that defines the form are defined. When we consider the space of newforms in $S_k(N)$ of weight k and level N, we say that its degree p corresponds to its finite field of definition $\mathbb{F}_p$. We observed that, for example, if $\dim S_k(N) = 1$, then the field of definition is $\mathbb{Q}$. More details on the field of definition are provided by Shimura [19].

2.6 Elliptic Curves

As mentioned in the introduction, elliptic curves are a type of object that have interesting properties that are studied in many areas of mathematics. In the next chapter, we will define L-functions that are associated with modular forms. In particular, we are interested in elliptic curves since, just like modular forms, they too have L-functions associated with them. In fact, there exists a correspondence between elliptic

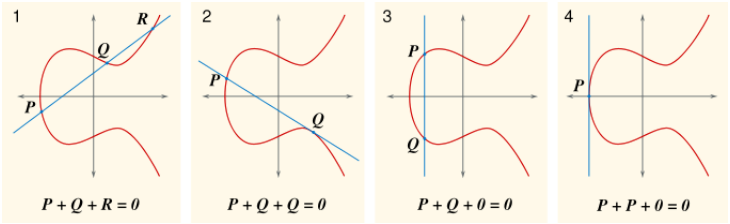


Figure 2.1: Elliptic Curve Point Addition

curves and modular forms which allows us to regard the L-functions of elliptic curves as L-functions of modular forms in our computations. Most of the information in this chapter can be referenced in Neal Koblitz's graduate text *Introduction to Elliptic Curves and Modular Forms* [12].

By definition, an elliptic curve is a curve with an equation of the form

$$y^2 = x^3 + Ax + B,$$

where the discriminant $4A^3 + 27B^2 \neq 0$. We denote the set of points of an elliptic curve over a field $\mathbb{F}$ as

$$E = \{(x, y) \in \mathbb{F} : y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\}.$$

We note that the extra point $\mathcal{O}$ refers to the point at infinity.

The necessity of the inclusion of the point at infinity can be seen in the geometric representations of elliptic curves. When we plot an elliptic curve, it appears to be a nonintersecting curve on $\mathbb{R}^2$. One interesting geometric property of elliptic curves is the addition of two points. To add two points P and Q on an elliptic curve, we draw a line through P and Q, and find the other point R where this line intersects the curve. If we then reflect R across the x-axis, the resulting point is $P + Q$. We see this illustrated in the first two images in Figure 2.1 [1].

Similarly, if we wish to add P to itself, we take the tangent line of P and find the point R where the line intersects the curve. By reflecting this point over the x-axis, we result in the point $2P$. This idea can be seen in the second two images in Figure 2.1.

The problem that arises with this geometric representation of addition on elliptic curves is what happens when the line we draw through either two points P and Q or

tangent to the point P is a vertical line. A vertical line will never intersect the elliptic curve at more than two points. Therefore, the addtion of the point $\mathcal{O}$ at infinity is necessary to preserve the additive geometric properties of elliptic curves. We say that $\mathcal{O}$ is the point that is found when traversing any vertical line towards infinity.

Algebraically, it can be proven that the solutions to an elliptic curve form an abelian group. Understanding the rational solutions

$$E(\mathbb{Q}) = \{(x, y) \in \mathbb{Q} : y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\}$$

is a long-standing Diophantine problem. This problem has been studied by exploiting the connections between modular forms, elliptic curves, and L-functions (which we define in the next chapter).

Chapter 3

L-functions

3.1 Defining L-functions

In this chapter, we will finally define the object which we wish to investigate in our experiments. The information following parallels work done in by Christoph Schmitt in his Master's thesis [18]. As we saw in Chapter 1, the definition of a modular form is abstract enough that it appears that there would be a number of ways to analyze its properties. One type of object associated with a particular modular form is an L-function. We define an L-function associated with a cusp form as follows:

Definition 7. *Let $f \in S_k(N)$. Then the L-function associated with f is the Dirichlet series*

$$L(s, f) = \sum_{n=1}^{\infty} \frac{a_n}{n^s},$$

which converges absolutely in the complex half-plane Re $s > k/2 + 1$.

We observe that the significance of a Dirichlet series is that it is of the form $\sum_{n=1}^{\infty} \frac{a_n}{n^s}$ with (a_n) a sequence of complex numbers for any s within the defined domain. In our case, the selection of the domain is simpler since we are considering s as a complex number based on the weight k of f and (a_n) as a real sequence [19].

3.2 Finding the Functional Equation

For a particular f, we analytically continue the series $L(s, f)$ through the use of the Mellin transform. The Mellin transform is an integral transform that may be regarded as the multiplicative version of the two-sided Laplace transform. It is defined by the integral

$$M(s, f) := \int_0^\infty f(iz) z^{s-1} dz.$$

Now we compute for our $f \in S_k(N)$: for s such that $\mathrm{Re}(s) > \frac{k}{2} + 1$,

$$\begin{aligned} M(s, f) &= \int_0^\infty f(iz) z^{s-1} dz \\ &= \int_0^\infty \sum_{n=1}^\infty a_n e^{-2\pi n z} z^{s-1} dz \\ &= \sum_{n=1}^\infty a_n \int_0^\infty e^{-t} t^{s-1} (2\pi n)^{-s} dt \\ &= \Gamma(s)(2\pi)^{-s} \sum_{n=1}^\infty \frac{a_n}{n^s} \\ &= \Gamma(s)(2\pi)^{-s} L(s, f), \end{aligned}$$

where $\Gamma(x) = \int_0^\infty x^{s-1} e^{-x} dx$ is the Gamma function. We note that the interchange of summation and integration included in the above computation is valid over the region of absolute convergence we are considering.

We now wish to be more specific in finding the so-called functional equation for Hecke eigenforms in $S_k^\pm(N)$. We recall from Theorem 1 that the spaces S_k^+ and S_k^- span the space $S_k(N)$, and that $f \in S_k^\pm(N)$ satisfies

$$f\left(\frac{-1}{Nz}\right) = \mu N^{k/2} z^k f(z)$$

where μ is the Atkin-Lehner eigenvalue of f. We will now see why the Atkin-Lehner eigenvalue of a cusp form is essential to finding the functional equation of its associated L-function.

We consider our computation of the general Mellin transform of f. This time, instead of simply replacing f by its Fourier series, we split the integral along the positive real axis at $\sqrt{N}$. In this way, we intend to find an analytic continuation of $L(s, f)$ to the entire complex plane. We compute

$$\begin{aligned} M(s,f) &= \int_0^\infty f(iz)z^{s-1}dz \\ &= \int_{\frac{1}{\sqrt{N}}}^\infty f(iz)z^{s-1}dz + \int_0^{\frac{1}{\sqrt{N}}} f(iz)z^{s-1}dz \\ &= \int_{\frac{1}{\sqrt{N}}}^\infty f(iz)z^{s-1}dz + \int_{\frac{1}{\sqrt{N}}}^\infty f\left(\frac{1}{Nz}\right)\left(\frac{1}{Nz}\right)^{s-1}\frac{1}{Nz^2}dz \\ &= \int_{\frac{1}{\sqrt{N}}}^\infty f(iz)z^{s-1}dz + N^{k/2-s}\mu i^k \int_{\frac{1}{\sqrt{N}}}^\infty f(iz)z^{k-s-1}dz. \end{aligned}$$

If we recall our intial definition of the Fourier series of f, we see that $f(iz)$ decays exponentially as $z \to \infty$. Then both of the integrals in the above sum converge for all $s \in \mathbb{C}$. Thus, we have found a way to extend our definition of $M(s, f)$, and thus $L(s, f)$, to cover the entire complex plane.

We also consider the Mellin transform when we replace s with $k - s$. This substitution results in the equation

$$M(k-s,f) = \int_{\frac{1}{\sqrt{N}}}^\infty f(iz)z^{k-s-1}dz + N^{s-k/2}\mu i^k \int_{\frac{1}{\sqrt{N}}}^\infty f(iz)z^{s-1}dz.$$

If we compare the resulting formulas for $M(s, f)$ and $M(k - s, f)$, with simplification we end up with in the equality

$$M(k-s,f) = N^{s-k/2}\mu i^k M(s,f).$$

We note that $\mu i^k = \pm 1$, since we recall from Proposition 3 that the weight k of f is always an even integer. To complete our asymmetric functional equation of $L(s, f)$, we substitute in our orginal computations for $M(s, f)$ to the above equation and simplify to result in

$$N^{s/2}(2\pi)^{-s}L(s,f) = \mu i^k N^{(k-s)/2}(2\pi)^{s-k}\Gamma(k-s)L(k-s,f).$$

This will be useful to us in our computations, since now we will be able to compute values of $L(s, f)$ for a particular f without the restriction of having to consider s such that $\mathrm{Re}(s) > \frac{k}{2} + 1$.

3.3 The Riemann Zeta Hypothesis and the Critical Line

Now, we will take a brief detour to discuss the Riemann zeta function. We do this to describe the Riemann Hypothesis, a conjecture which relates to the experimental computations of L-functions we will carry out.

The Riemann ζ-function, $\zeta(s)$, is a function of a complex variable s that analytically continues the sum of the infinite series $\sum_{n=1}^{\infty} \frac{1}{n^s}$, which converges when the real part of s is greater than 1 [17].In fact, it is easy to see that the Dirichlet series that we considered in defining L-functions of modular forms is simply a generalization of the Riemann zeta function.

One interesting property of the Riemann ζ-function that was discovered by Euler is the connection between the zeta function and the prime numbers.

Theorem 7. *For all prime numbers p,*

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p\, prime} \frac{1}{1 - p^{-s}}$$

where both sides of the equation converge for Re(s) > 1.

The product on the right side of the equation in the above formula is known as an *Euler product*.

Also, from Riemann's orginal paper [17], the Riemann zeta function is known to satisfy the functional equation

$$\pi^{-s/2}\Gamma\left(\frac{s}{2}\right)\zeta(s) = \pi^{-\frac{1-s}{2}}\Gamma\left(\frac{1-s}{2}\right)\zeta(1-s).$$

In the study of L-functions of cusp forms, an important conjecture related to the Riemann zeta function is known as the Riemann Hypothesis. Though as a conjecture

it has never been proved, it is so widely supported by experimental evidence that it and its related results about L-functions associated with cusp forms in general are commonly assumed to be true in most experiments involving the computation of zeros of L-functions.

The Riemann Hypothesis, proposed by Bernhard Reimann in 1859, is stated as follows: *All non-trivial zeros of the Riemann zeta function have real part* $\frac{1}{2}$. In this case, non-trivial refers to zeros that do not arise from poles of the Gamma factor. If the hypothesis is correct, then all non-trivial zeros of the function lie on the *critical line* $\mathrm{Re}(s) = \frac{1}{2}$.

For our purposes, we recognize that like the ζ-function, L-functions associated with cusp forms are primarily defined by a Dirichlet series. They also satisfy a functional equation which appears very similar to that of the zeta function.

We recall (see Theorem 2) that one of the important properties of Hecke eigenforms in particular is that their coefficients are multiplicative. This means that it is possible to create an analogous Euler product formula for $L(s, f)$

$$L(s,f) = \prod_{p|N} \frac{1}{1 - \frac{a_p}{p^s}} \prod_{p \not| N} \frac{1}{1 - \frac{a_p}{p^s} - \frac{1}{p^{2s+1-k}}}$$

which converges for Re $s > \frac{k+1}{2}$.

Due to these similarities, it is possible for us to generalize the Riemann hypothesis to be a conjecture about L-functions associated with newforms as follows: *All non-trivial zeros of an L-function associated with a newform of weight k and level N have real part* $\frac{k}{2}$.

What we have observed is that the Riemann zeta function and its associated Hypothesis is merely a special case which illustrates an important assumption behind our computations of zeros of L-functions: all zeros of an L-function associated with a newform f can be found on the critical line $\mathrm{Re}(s) = \frac{k}{2}$.

Currently our Euler product formula for L-functions associated with newforms implies that $L(s, f)$ does not vanish for Re $s > \frac{k+1}{2}$. We define the *critical strip* to be the set $\{s \in \mathbb{C} : \frac{k-1}{2} \leq \mathrm{Re}\ s \leq \frac{k+1}{2}\}$, and we note that the Dirichlet series of an L-function is not convergent in the critical strip. Thus, we have a way to perform calculations with Re $s > \frac{k+1}{2}$, and by a non-trivial result of Rankin-Selberg theory [8], we have that $L(s, f)$ is non-vanishing at Re $s = \frac{k+1}{2}$ as well.

However, we still require a way to perform calculations for Re $s < \frac{k+1}{2}$. To do this, we will investigate the *approximate functional equation*.

Theorem 8. *Let $f \in S_k(N)$ with Atkin-Lehner eigenvalue μ. Then the L-function L(s,f) associated with f satisfies the approximate functional equation*

$$L(s,f) = \frac{1}{\Gamma(s)} \sum_{n=1}^{\infty} \frac{a_n}{n^s} \Gamma\left(s, \frac{2\pi n r}{\sqrt{N}}\right) + \mu i^k N^{k/2-s} (2\pi)^{2s-k} \frac{1}{\Gamma(s)} \sum_{n=1}^{\infty} \frac{a_n}{n^{k-s}} \Gamma\left(k-s, \frac{2\pi n}{\sqrt{N} r}\right).$$

We note that the convergence of this representation implies the analytic continuation of $L(s,f)$. In order to perform computations using this formula, we perform truncated estimates on the two infinite series within the terms of the equation

Often when people investigate the Riemann zeta function, they find it is useful to investigate the function

$$\xi(s) = \pi^{s/2} \Gamma\left(\frac{s}{2}\right) \zeta(s),$$

since ξ has the functional equation $\xi(s) = \xi(s-1)$.

When we generalize this concept to L-function we define the equation

$$\Lambda(s,f) := N^{s/2} (2\pi)^{-s} \Gamma(s) L(s,f),$$

which satisfies the functional equation

$$\Lambda(s,f) = \mu i^k \Lambda(k-s, f).$$

Similar to $\zeta(s)$, this is a convenient and symmetric functional equation.

3.4 Analytic Normalization

In our above computations, we observe that the critical line on which the zeros of the L-function fall is determined based on the weight of the modular form to which the L-function is associated. In our computations, we wish to avoid these differences in the critical line, so as to more easily compare the statistics of the zeros on the critical line of our various L-functions. To do this, we are going to analytically normalize the coefficients of each modular form so in all cases the critical line becomes $\mathrm{Re}(s) = \frac{1}{2}$.

To do this, we apply a change of variables to each of the coeficients in the Fourier series of a modular form, substituting $s - \frac{k-1}{2}$ in for s. This results in the following formula for the series

$$\sum \frac{a_n / n^{\frac{k-1}{2}}}{n^s}.$$

By applying this change of variables, we result in the formula

$$\Lambda(s, f) = \mu i^k \Lambda(1 - s, f),$$

which normalizes the coefficients of the L-function associated to a modular form. This normalization also allows us to avoid any computational errors that might occur from attempting to compute using extremely large coefficients.

3.5 The Hardy Z-Function

Now that we have a general approximate functional equation for L-functions of analytically normalized modular forms, it would be beneficial if we could use this equation over the domain of the entire complex plane. In order to achieve this result, we use a tool known as the *Hardy Z-function*. For the Riemann ζ-function, we define the Hardy Z-function by

$$Z(t) = e^{2\pi i t} \zeta\left(\frac{1}{2} + it\right).$$

It follows from the functional equation of the Riemann ζ-function that the Hardy Z-function is real for all real values of t. Moreover, it follows that the non-trivial zeros of $Z(t)$ are precisely the zeros of the ζ-function along $s = \frac{1}{2}$ [18].

In our case, we use a general form for the Hardy Z-function which holds over for not just the Riemann ζ-function, but for all L-functions of modular forms. This allows us to define a real and differentiable representation of L on the critical line that shares the same zeros as $L(\frac{1}{2}, f)$. For a modular form f, we define the Hardy Z-function to be

$$Z(t) = \frac{\mathrm{Re}\left(\Lambda(\frac{1}{2} + it, f)\right)}{|\Gamma(\frac{1}{2} + it, f)|} \left(\frac{2\pi}{\sqrt{N}}\right)^{1/2}.$$

We note that when $\mu i^k = -1$, we have a vanishing real part of Λ on the critical strip. Hence in that case we define $Z(t)$ by replacing Re Λ with Im Λ. Since the non-trivial zeros of this Z-function are precisely the non-trivial zeros of the L-function along $s = \frac{1}{2}$, we now apply the techniques known for solving for zeros of the Hardy Z-function in order to determine zeros of the corresponding L-function on $\mathrm{Re}(s) = \frac{1}{2}$. In this way, we now have a way to solve for zeros of an analytically normalized L-function associated with a modular form over the entire complex plane.

3.6 Order of Vanishing at $s = \frac{1}{2}$

Finally, we note that so far we have defined modular forms based on two parameters: weight k and level N. One other characteristic of L-functions that will be important to us in our computations is known as the *rank*, r, of an L-function associated with a modular form. Intuitively, the rank of a modular form is related to the *order of vanishing* of the L-function at $s = \frac{1}{2}$. The Birch-Swinnerton-Dyer conjecture [24], another conjecture which is widely supported by experimental evidence, though yet to be proved, says as follows:*the rank of an elliptic curve L-function is the order of the zero of its associated L-function at the central point* $s = \frac{1}{2}$. For example, if the first zero on the critical line does not occur at the central point, we can say that the newform has rank 0.

As rank increases above 0, however, other factors besides the value of the first zero on the critical line come into play, making us unable to simply assume the value of the rank based on the order of the first zero with a nonzero y-value. Therefore, more complex ways of precisely computing the exact rank of a newform have since been found [20]. One important tool which we use in our computations is the sign of the functional equation of an L-function, denoted by ω. If ω is positive, we know we have an even function, and if ω is negative, we know we have an odd function. Thus, by using a combination of the BSD conjecture, the value of the first zero on the critical line, and the sign of the functional equation, we can almost always determine the rank of an L-function computationally.

3.7 Conductor of an L-Function

In this section, we define a characteristic of an L-function that plays a key role in the random matrix theory correspondence we will describe in the next chapter. This characteristic is known as the *conductor* of an L-function. Following the results of Booker [6], we define the *analytic conductor* of an L-function to be

$$Q(s) = N \sum_{j=1}^{k} \frac{s + \mu_j}{2\pi}$$

where N is the level of the modular form and k is the weight. This function provides us with a tool to induce a partial ordering on a set of L-functions, which will be usual for us when we wish to compare L-functions in our computations.

3.8 L-functions of Elliptic Curves

One technique used to understand solutions over the rationals is to consider solutions modulo p, where p is a prime number. In our case, we will be considering $E(\mathbb{F}_p)$ where $\mathbb{F}_p$ is the finite field of the integers modulo p. Then for $f(x)$ a cubic polynomial, we can define

$$E(\mathbb{F}_p) = \{(x_0, y_0) \in \mathbb{F}_p^2 : y_0^2 = f(x_0)\}.$$

Then the L-function associated with E is the Dirichlet series

$$L(E, s) = \sum_{n=1}^{\infty} \frac{a_n}{n^s}$$

which converges absolutely in the complex half-plane Re $s > 3/2$.

The following result about elliptic curves follows from the proof of Fermat's Last Theorem, due to Wiles, Taylor-Wiles, and Breuil-Conrad-Diamond-Taylor [7], and summarizes an important connection between modular forms and elliptic curves.

Theorem 9. *Let $E(\mathbb{F}_p)$ be the solutions to an elliptic curve $y^2 = f(x)$ where $f(x)$ is a cubic polynomial and p is prime. Then if we define*

$$a_p = p + 1 - |E(\mathbb{F}_p)|$$

and $a_{mn} = a_m a_n$ for m, n relatively prime, the resulting series

$$f(z) = \sum_{n=1}^{\infty} a_n q^n$$

with $q = e^{2\pi iz}$ is a modular form of weight 2 and level N, where in this case we refer to N as the conductor of the elliptic curve. Moreover, and this is key, $L(f, s) = L(E, s)$.

We thus see that one established property of elliptic curves is that they correspond to a subset of modular forms of weight 2. We now recall our work in Chapter 3 on L-functions of newforms. We note that since all elliptic curves have weight 2, their associated L-functions derived using the Mellin transform are convergent on the half-plane Re $s > 3/2$. Furthermore, we recall that we are able to find an analytic continuation of the L-function that satisfies the functional equation

$$\Lambda(s, f) = -\mu\Lambda(2 - s, f).$$

3.8.1 The Excess Rank Phenomenon of Elliptic Curves

Another important characteristic about elliptic curves in particular that has been observed in computations involving L-functions associated with curves is known as the *excess rank phenomenon*. The set of all elliptic curves over $\mathbb{Q}$ has approximately half of its functional equations even ($+1$ in the functional equation) and half odd (-1 in the functional equation). We recall from Section 3.6 the Birch and Swinnerton-Dyer conjecture, which states that the rank of a newform is directly related to the order of the zeros of its L-function. We consider a family of elliptic curves over $\mathbb{Q}$ and apply the Birch and Swinnerton-Dyer conjecture. With the application of something known as the Density Conjecture, we arrive at the excess rank phenomenon: *at the central point in the limit as the conductors tend to infinity the L-functions have rank 0 half of the time and rank 1 half of the time*. Therefore, as we increase the conductor of an elliptic curve towards infinity, we observe that the existence of curves of rank $r \geq 2$ becoming less likely.

We pause here to note the overall significance of the subset of elliptic curves within the overall context of modular forms. Due to the much more specific properties of elliptic curves, much more research and analysis has been put into some of the phenomena observed regarding the L-functions associated with them. Also, for reasons we will explain in Chapter 5, computations involving elliptic curves are much less time

intensive compared to analogous computations over modular forms, especially as we increase the condutor of an elliptic curves (i.e. the level of a modular form). Therefore, much of the theory and experimentation done in the following chapters has only been observed for L-functions of elliptic curves, and even in that case, L-functions of specific families of elliptic curves. Keeping this in mind, we will proceed to show some of the previous theory and experimentation regarding L-functions associated with families of elliptic curves, and how we have been able to extend those results over L-functions associated with all elliptic curves, and even those associated more generally with any eigenform.

Chapter 4

Random Matrix Theory

4.1 Historical Background

Random matrix theory is a field of mathematics which gained momentum beginning in the 1950s based on work done by Eugene Wigner in mathematical physics. In his work, Wigner wanted to find a way of describing the properties of the spacings between energy levels of heavy nuclei in highly excited states (such as those measured in nuclear reactions). These complex systems of energy levels of nuclei live in an infinite-dimensional Hilbert space. Therefore, Wigner conjectured that the energy levels could be approximated by the eigenvalues of a very large random matrix. More specifically, the spacings between energy levels of these such nuclei could be modelled by the spacings between successive eigenvalues of a random $(N \times N)$- matrix as $N \to \infty$ [22].

In April 1972, a large advance was made in connecting number theory to random matrix theory based on a chance encounter of Hugh Montgomergy and Freeman Dyson. At the time, Montgomergy was visiting Princeton to discuss his work on the nontrivial zeros of the Riemann zeta function with a number theorist interested in their connection to the pattern of the prime numbers. In what became known as the Pair Correlation Conjecture, Montgomery proposed that the zeros of the function appeared to repel other nearby zeros. During teatime at Princeton, Montgomery mentioned this result to Freeman Dyson [22].

Dyson, coincidentally, had been one of the key collaborators with Wigner in his work on the statistical behavior of the eigenvalues of random matrices. He immediately discovered an amazing heuristic correspondence between Montgomergy's recent results and the observations he had made about the eigenvalues of random matrices with Wigner years earlier. Today, more than forty years later, the answer as to why there is such a correspondence between the two statistical distributions has prompted much work in intersecting areas of number theory, mathematical physics, probability, and statistics. Overall, there's no explanation for why there should be such a heuristic correspondence.

However, there has been some other work besides the "Princeton Tea" that indicates that this correspondence should exist. Years before, Hilbert and Polya stated a conjecture indicating that the Riemann Hypothesis is true because non-trivial zeros of the zeta function correspond to the eigenvalues of some positive operator [22]. At the time, there was little basis for the conjecture. Montgomery's conclusions in 1972 provided a more solid basis for the conjecture, though obviously did not actually prove it true. More recently, Katz and Sarnak have proved the desired results of a correspondence between random matrix theory and L-functions over function fields [22]. However, the connection still remains unproven for L-functions over all of $\mathbb{Q}$, which is the correspondence we are investigating.

4.2 The Basics: What is Random Matrix Theory?

Before delving into a more detailed explanation of the particular random matrix theory and number theory connection we are interested in, we provide a brief introduction to field of random matrix theory itself. Random matrix theory involves the study of collections of "random" sets of $N \times N$ matrices. The matrices are referred to as "random" since their entries are independently chosen from a fixed probability distribution. We call a collection of matrices, along with the probability density which describes how likely it is to observe a given matrix a *random matrix ensemble*.

However, there is another notion of randomness that has been more useful for number theory since it leads to a more natural method of choosing a matrix at random.

Definition 8. *We say that a complex square matrix U is unitary if*

$$U^*U = UU^* = I$$

where I is the identity matrix and U^* is the conjugate transpose of U.

Let $U(N)$ be the space of $N \times N$ unitary matrices, and consider its compact subgroups. For example, one commonly studied compact subgroup of $U(N)$ is the $N \times N$ orthogonal matrices. For each one of these subgroups, there exists a measure $O(n)$ known as the *Haar measure* attached to it. This measure can be used to find a random matrix ensemble which is useful for number theory. We also note that the eigenvalues of unitary matrices have modulus 1, and can be written as $e^{i\theta_j}$ with each θ_j real. Then the θ_j's provide us with a sequence of real numbers which can be statistically analyzed.

The main idea used to learn more about the eigenvalues of matrices in random matrix theory is known as the *Eigenvalue Trace Formula*. We first recall that the trace of a matrix A is the sum of its diagonal entries:

$$\text{Trace}(A) = a_{11} + ... + a_{NN}.$$

We generalize this idea when we apply the idea of the trace to the power of matrices.

Theorem 10. *Let A be an $N \times N$ matrix. Then*

$$\text{Trace}(A^k) = \sum_{i_1=1}^{N} ... \sum_{i_k=1}^{N} a_{i_1 i_2} a_{i_2 i_3} ... a_{i_{k-1} i_k} a_{i_k i_1}.$$

We use the equivalence from the theorem above to help us in understanding the Eigenvalue Trace Formula, which is stated as follows: for any non-negative integer k, if A is an $N \times N$ diagonalizable matrix with eigenvalues $\lambda_i(A)$, then

$$\text{Trace}(A^k) = \sum_{i=1}^{N} \lambda_i(A)^k.$$

The importance of this formula is that it provides a relationship between the randomly chosen entries of a matrix and the eigenvalues of that matrix. This is key, since the eigenvalues of the matrices in a random matrix ensemble are the objects around which the field of study is based. In fact, the Eigenvalue Trace Formula is in effect the idea that makes the entire field of random matrix theory possible.

Finally, we make note of the idea of normalization in terms of eigenvalues of random matrices. Similarly to how we normalized the coefficients of the Fourier

series of modular forms in order to be able to compare the zeros of their associated L-functions, it is also important to normalize the eigenvalues of random matrices to be able refer to them on a similar scale. For the details of this normalization process, we refer to Miller's *An Invitation to Modern Number Theory* [15].

4.3 Generating Random Matrices From Classically Compact Groups

In the above section, we mentioned that random matrices come from the compact subgroups of the space of unitary $N \times N$ matrices. In this section, we will describe the sort of algorithm that is used by mathematicians to construct these random matrices.

We start by noting that the columns of an $N \times N$ unitary matrix are orthonormal vectors in $\mathbb{C}^N$. We recall the process of Gram-Schmidt orthonormalization from linear algebra, and see that if we take a complex $N \times N$ matrix Z of full rank and apply Gram-Schmidt orthonormalization to its columns, the resulting matrix Q is unitary. Then we result in $Z = QR$ where R is upper-triangular and invertible. This is known as the *QR decomposition*, and is a factorization that is widely used in numerical analysis.

We note that the QR decomposition is not a unique map. Instead, it defines

$$QR : GL(N, \mathbb{C}) \to U(N) \times T(N)$$

for $Z \in GL(N, \mathbb{C})$ and $T(N)$ the group of invertible upper-triangular matrices. Therefore, we require some way to alter the QR factorization to make a one-to-one map. We consider the one-to-one map

$$\overline{QR} : GL(N, \mathbb{C}) \to U(N) \times T(N)/(T(N) \cap U(N))$$

where $T(N)/(T(N) \cap U(N))$ is the right coset space of $T(N) \cap U(N)$ in $T(N)$. This map is constructed in order to induce the Haar measure on $U(N)$, which from the previous section we know is a necessary condition to find our random matrix ensemble.

We now have all the necessary tools which will allow us to create a random unitary matrix with distribution given by the Haar measure. First, take any $N \times N$ complex matrix Z whose entries are complex standard normal random variables. Then run Z through any QR decomposition. Next, create the diagonal matrix D with diagonal

entries $\frac{r_{jj}}{|r_{jj}|}$ where the r_{jj}s are the diagonal elements of R. Finally, we note that the diagonal elements of $R' = D^{-1}R$ are always real and strictly positive. Thus we finish by taking $Q' = QD$, which is therefore distributed with Haar measure as required. A more detailed description of the results of this section can be found in Mezzadri's paper [13].

4.4 Random Matrix Theory and Number Theory Connection

We now return to the Princeton teatime discovery by Dyson and Montgomery of a heuristic correspondence between the spacing of zeros on the critical line of L-functions of modular forms and the spacing between eigenvalues of random matrices. In our experiments, we explore another connection between L-functions and random matrix theory. This connection examines the heuristic correspondence between the effect of multiple zeros at the central point on nearby zeros of an L-function and the effect of multiple eigenvalues at 1 on eigenvalues in a classical compact group. This correspondence is detailed in Katz and Sarnak's Density Conjecture, which states that the behavior of the normalized zeros of families of L-function of modular forms near the central point as the levels tend to infinity is equal to the $N \to \infty$ scaling limit of the normalized eigenvalues near 1 of a $(N \times N)$ classical compact group [11].

In stating the last conjecture, we notice that we referred to something known *families* of L-functions. In fact, there is no universally agreed upon definition for families of L-functions. In loose terms, however, we can define a family of L-functions as some set of L-functions that has an ordering on it. In particular, when we are referring to L-functions, we often order them based on what is known as the *conductor* of the L-function, which we defined in Section 3.7. We note that according to that formulaic definition, we can therefore order our families of L-functions associated with modular forms based on either the level N or the weight k of the forms while holding the other parameter constant.

While the theory behind which particular classical compact group has normalized eigenvalues corresponding to the zeros at the central point of a certain family of L-functions, it is the intuitive idea of a correspondence at the limit of $N \to \infty$ which is important to our experiments. At Williams College, work is currently being done by Miller and his students to prove the random matrix theory hypothesis which

states that the attraction observed between eigenvalues near 1 of a classical compact group is maximized as $N \to \infty$. In our experiments, we address the other side of this correspondence: if we know that as N goes toward infinity the attraction is maximized, what can we observe about the attraction at values of $N < \infty$?

Chapter 5

Experimental Computations

5.1 Modular Forms in Sage

In our experiments, we performed computations involving modular forms using Sage [5], a mathematical Python library. Within Sage, we are able to create modular form objects defined by the weight and level of the form. Once we have created a particular modular form object, there are then a few main computations we need to perform in order to find the L-function of that modular form. Unfortunately, some of these computations are more computationally expensive than others.

One attribute of the modular form that we need to compute is the Atkin-Lehner eigenvalue. If we recall the ideas from Section 2.2, we can see intuitively that this computation should run fairly quickly when done using a computer; indeed, it is not a very time-intensive computation no matter the level or weight of a form. We also need to consider the computation of the coefficients of the L-function associated to a particular form. In fact, this does get slower as we increase the level and/or weight, since the dimension of the space $S_k(N)$ gets large.

Finally, we must consider the way in which we compute the entire set of Hecke eigenforms of a particular level and weight. To do this, we create a newforms object based on the parameters level and weight, and assign a to it modular symbol, which is the way Sage represents a modular form. The modular symbol has a field of definition of degree n, and so for any modular symbol α, we receive α and its n representations

back from Sage.

However, computation of the coefficients is eventually necessary in order to find the L-functions associated with the newforms. This computation is where we can run into issues of efficiency based on the size of the level and weight of a modular form. Let $f \in S_k(N)$. Then by a theorem from Shimura [19], the coefficients of f live in a totally real field K_f. We need to do exact linear algebra in this field, the degree of which grows as the level and/or weight increases. Therefore, it is quickly impossible for us to perform these linear algebra computations as the level and/or weight of the modular forms associated with L-functions tend towards infinity.

5.2 Elliptic Curves in Sage

Within Sage, the elliptic curve object differs significantly from that of a general modular form. The curve is created based on its conductor. However, for most conductors there are multiple elliptic curves, many of which have different corresponding L-functions. Therefore, Sage differentiates between curves with the same conductor by attaching a letter onto the name; e.g. 11a is the name of the first elliptic curve object with condutor 11. An illustration of this principle can be seen in the LMFDB database [3].

Once again, our computations require us to compute a large number of normalized coefficients of the L-function corresponding to a particular elliptic curves. However, there are two features of Sage which make computing L-functions of elliptic curves computationally much more efficient and less time-intensive than modular form computations. The first is what is known as the Schoof-Elkies-Atkin (SEA) algorithm [5]. The SEA algorithm is used for counting the number of points on an elliptic curve over a finite field, the a_p in the Euler product that defines $L(E, s)$. The algorithm is known to be sublinear, making it a computationally inexpensive implementation for the large number of calculations we made in our experiments [5].

The second useful aspect of the Sage program is that in it one can use the Cremona database. While the basic Sage package contains data on elliptic curves up to conductor $10,000$, the Cremona database is an add-on to a custom Sage installation which ranges up to curves of condutor $350,000$. This package therefore gave us access to perform an unprecedented number of calculations on millions of L-functions of different elliptic curves, all of which were able to be done at the same computationally

efficient rate implemented as in the lower conductor ranges.

5.3 L-function Calculations

Calculations involving L-functions for these experiments were done using the lcalc command line interface for computing with L-functions developed by Michael Rubinstein [2]. The lcalc interface he developed requires the creation of a text file containing defining data associated with a particular modular form, such as its level and coefficients. The program then allows for computationally inexpensive calculations involving a L-function represented by one of these such files. These calculations are done via an optimized version of the Approximate Functional Equation (see Theorem 4). Some important calculations that can be completed through the use of the program include computing the values of the first n zeros on the critical line or computing the value of the L-function at a particular point.

5.4 Challenges in Implementation of Code

The basic workflow implementation for the code we wrote can be seen in Figure 5.1. In the figure, we see that we began by creating either an elliptic curve or modular form object, a process which is described in Sections 5.1 and 5.2. From there, we found the data of the L-function associated with the object, a process which is described in Section 5.4.2. Then, we wrote this data into a file of the form required by Rubinstein's lcalc program. Finally, our last step was to create shell files which would run the L-function file through lcalc to output a file containing the data we desired for that L-function. This entire workflow had to be repeated for each L-function we considered, so finding a way of optimizing this process for each family of L-functions was a very important part of our research.

While working out the details of this workflow, we encountered many standard coding challenges that were easily fixed. However, as we were implementing the workflow, the creation of the code in order to calculate the data relevant to our research also required solutions to three distinct challenges:

1. How can we find all of the modular forms within a particular family we have

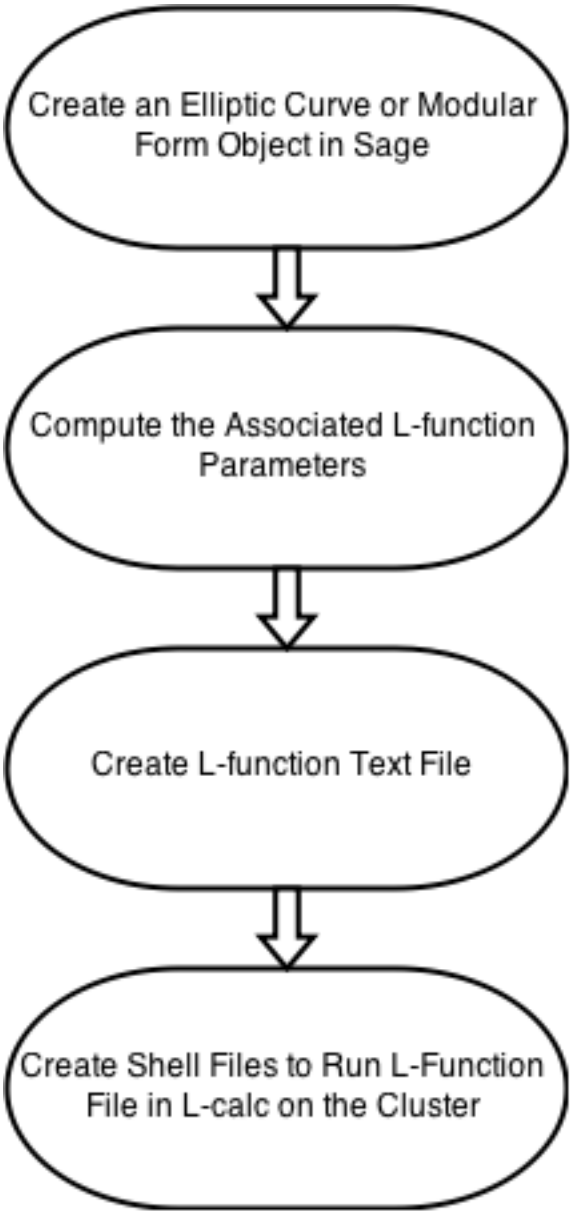


Figure 5.1: Code Workflow

defined?

2. Given a family of elliptic curves or modular forms, how can we efficiently compute the defining characteristics of the L-functions associated with each of the forms?

3. How can we efficiently compute the statistical data relevant to our research once we have the defining characteristics of particular L-functions?

The code described in this section can be found in Appendix 1.

5.4.1 Finding Families of Modular Forms

The first question which we were needed to consider when designing our experiments was on which "families" of modular forms we wished to perform our data analysis. The definition of a family of modular forms can vary widely in specificity depending on the source; in our case, however, we consider a family to be the set of forms created by the fixing of one parameter, such as level or weight, and the varying of the other (in some specific way).

One common way of creating a family of this type is the process known as *twisting*. In this technique, we begin with a base L-function associated to some elliptic curve or modular forms. We then vary the condutor of the L-function by multiplying by a fundamental discriminant d. This means that $d \equiv 1 \pmod 4$ and is square-free, a calculation which can be done quickly to result in a long list of eligible d's. Using this list, we are able to create an entire family of L-functions up to some conductor.

In our experiments, due to computational efficiency concerns later on in the process, we began by creating families of elliptic curves. Since elliptic curves correspond to modular forms of weight 2, by considering curves we were in effect fixing a certain weight. Therefore, by choosing a base curve of a certain conductor, we were then able to create a family of twists of that curve by using the twisting process just described. In this way, we were able to efficiently create hundreds of families of twists of elliptic curves on which we could perform further calculations. To do this in Sage, we first wrote code to create elliptic curve objects based on conductor. We then wrote code that found a large list of eligible d's with which to twist our L-function.

From there, we considered a more comprehensive family of modular forms: the

family of all elliptic curves up to some conductor. We recall from Theorem 9, since we are considering just elliptic curves, we are in effect fixing the weight at 2. By considering the family of all elliptic curves, we are simply fixing the weight and varying the conductor in as many iterations as possible. In order to ensure that we cover all possible conductor variations, we looped through the entire Cremona database, which as we stated above contains all elliptic curves up to conductor 350000. Therefore, we were able to create a large family of millions of curves simply by considering every possible elliptic curve in the database. To do this in Sage, we again used our code for creating elliptic curve objects based on conductor. However, this time we also had to write code to loop through the entire Cremona database.

Finally, we wished to consider families of general modular forms. To do this, we created families in two different ways: fixing the weight at 2 and varying the level, and fixing the level at 1 and varying the weight. By fixing such low initial values, we were attempting to avoid the computational efficiency concerns of later steps in the process for as long as possible. In fact, by considering these two particular families of modular forms, we actually covered a large subsection of the modular forms with associated L-functions that are computationally efficient to compute. To do this in Sage, we wrote code to create the basis of newforms of a particular weight and level. We then had to write code to create L-functions for each of the individual forms within the set of newforms.

5.4.2 Efficiently Computing L-Functions

To address the problem of computing the defining characteristics of L-functions associated with modular forms efficiently, we first began with our families of twists of elliptic curves. We began by computing the attributes of the L-function associated with our base curve. Computing the L-function of a modular form, however, takes time. Therefore, instead of continuing to compute L-functions for a series of elliptic curves, we instead not only twist the conductor, but we also twist the other characteristics of our orginal L-function in order to arrive at the characteristics of an entirely new L-function.

Twisting L-function data is computationally efficient since it requires fewer calculations than the computation of an entire elliptic curve and then its associated L-function from scratch. Consider a base L-function L with conductor N_L, sign ω_L, and coefficients $a_{n,L}$. We recall that to go from a base L-function L to the L-function of a twist $\tilde{L}$, we simply need to find a fundamental discriminant d. This means that

$d \equiv 1 \pmod 4$ and is square-free, a calculation which can be done quickly to result in an entire list of eligible d's. From there, the conductor of the twist is now found by $N_{\tilde{L}} = N_L d^2$. We then compute the Kronecker symbol $(d|N)$, which is simply a generalization of the Jacobi symbol. Using this symbol, we compute

$$\omega_{\tilde{L}} = \omega_L \frac{d}{|d|} * (d|N)$$

$$a_{n,\tilde{L}} = a_{n,L}(d|N)$$

for each a_n which we computed for our original L-function. Carrying over all of the other characterisitics of our original L-function to the new one, we have found an L-function contained within our family. By repeating this twisting computation on one L-function associated with a base elliptic curve for a series of d's, we get a family of twists of elliptic curves over which we can statisitically analyze L-function data. To do this in Sage, we first wrote code to compute the characteristics of the L-function associated with a particular elliptic curve object, and to write them into a file of the proper format necessary for lcalc. We then wrote code that could read in the L-function file of the base curve, perform the twisting computations on the characteristics, and write appropriate new L-function files for each twist.

We then considered our family of all elliptic curves. Since we were covering the entire family of curves in the Cremona database, it turned out that the most computationally efficient approach for finding the characteristics of the associated L-functions was simply to create each of the elliptic curve objects and find the associated L-function attributes directly from the objects. While this process was slightly more computationally costly than twisting L-functions, it was the only way for us to be able to find all of the L-functions associated with elliptic curves up to a certain conductor. Also, we note that this approach still was not overly computationally expensive, since as we established before, the computational efficiency of creating elliptic curve objects does not decrease significantly as the size of the conductor increases. To do this in Sage, we had to apply the code we had already written that computed the characteristics of the L-function associated with a particular elliptic curve object, and wrote them into a file of the proper format necessary for lcalc.

Finally, we considered our two families of modular forms. Once again, since we were considering an entire family of forms based on fixing one parameter and all possibilities of the other, our approach had to mirror the the one we took with the family of all elliptic curves. In this case, however, we ran into the problem of computational inefficiency once the parameter became too large, and therefore our families in these two cases are significantly smaller than our family of all elliptic

curves, in order to maintain computational efficiency as much as possible. To do this in Sage, we had to write code that computed the characteristics of the L-function associated with a particular general modular form object, and wrote them into a file of the proper format necessary for lcalc.

5.4.3 Computing Statistical Data on L-Functions

Once we had found all of the families of L-functions described in the following sections, our final problem to address was the most computationally efficient way to calculate the statistical data we needed for our data analysis we wished to conduct. At the time, the characteristics of each particular L-function could be found within a text file which needed to be passed through the lcalc interface we described earlier in the chapter. Therefore, we concluded that the most efficient way to perform these calculations was through the use of the Bucknell Linux Computing Cluster. By writing code that created a series of shell script files (see Appendix 2), we were able to run large amounts of lcalc calculations at the same time, resulting in a much more computationally efficient approach to acquiring the L-function data. Fortunately, by this point the process was identical for all of our families, since we had written all of the L-function files to be submitted to lcalc in the same format. To do this, therefore, we had to write code that created a shell job file of the proper format for each statistic about an L-function that we wanted to compute, and write these statistics into respective files. We also then had to write code of the correct format that ran all of these job files to the computing cluster. Finally, we had to write code to gather the statistical data from the respective files and perform different types of data analysis on the statistics, the results of which we will discuss in the next chapter. This code can be found in Appendix 3.

Chapter 6

Results, Conclusions, and Future Work

6.1 Prior Work

The first experiment of this kind was published by Miller in 2005 [14]. In his experiment, Miller discusses and formulates the conjecture that the zeros of L-functions with the parameters weight 2, level N (determined by some invarient of the curve $f(x) = y^2$), and trivial character are repulsed by each other. Figures 6.1 and 6.2 [14]are two histograms that demonstrate the results of Miller's initial experiment.

The first graph considers the data of the zeros of L-functions for smaller level N, while the second graph demonstrates the data for larger N. He observed that as the size of the conductor increases, the histograms for the experimental data on modular forms of weight 2 moves to the left. He compared these results to the random matrix theory results seen in Figure 6.3 [14], which indicates that the limit of the density should continue to move to the left.

In particular, Miller's experiments focused on elliptic curves of rank 0 and rank 2 (Figures 6.1 and 6.2 correspond to the results for rank 0). In both cases, Miller generated data on the first normalized zero above the central point for select families of elliptic curves with the same rank. He separated the data into two sections of the same size based on the size of the conductor of the curve. He then showed the

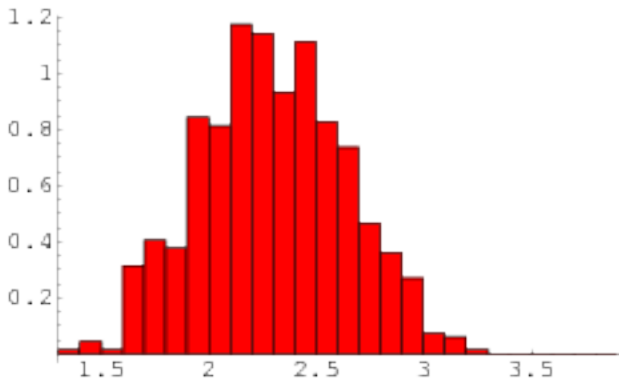


Figure 6.1: Miller Results for first normalized zero above the central point: 750 rank 0 curves from $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, log(cond) $\in$ [3.2, 12.6], median= 1.00, mean = 1.04, standard deviation above the mean= .32

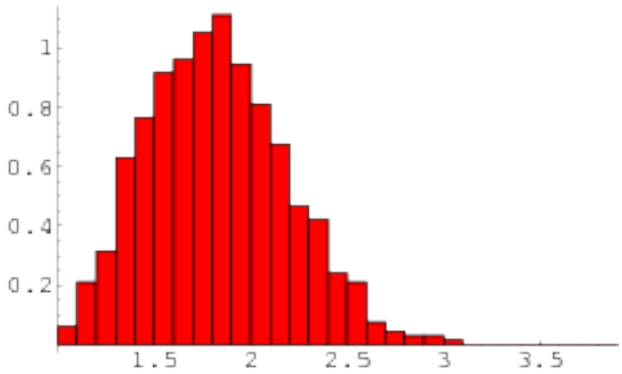


Figure 6.2: Miller Results forfirst normalized zero above the central point: 750 rank 0 curves from $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, log(cond) $\in$ [12.6, 14.9], median= .85, mean = .88, standard deviation above the mean= .27

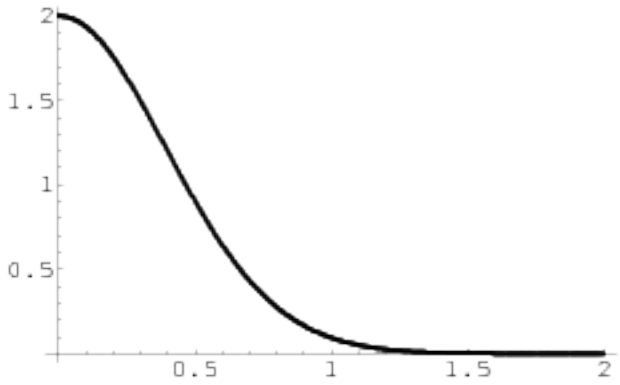


Figure 6.3: Random Matrix Theory Model: First normalized eigenangle above 1: $N \to \infty$ scaling limit

statistical relationship between the zeros within the two samples; namely, that the mean value of the zeros with smaller conductor is consistently significantly larger than the mean value of the zeros with larger conductor. For each family of elliptic curves that Miller analyzed, this relationship between conductor size and the mean value of the first zero above the central point remained consistent. This experimental evidence, he concluded, suggested that the first normalized zero is in some cases attracted to the central point. Also, as conductors increase, this attraction increases, verifying the theoretical random matrix theory results seen in Figure 6.3, which suggest the attraction should be maximized as conductor size tends to infinity.

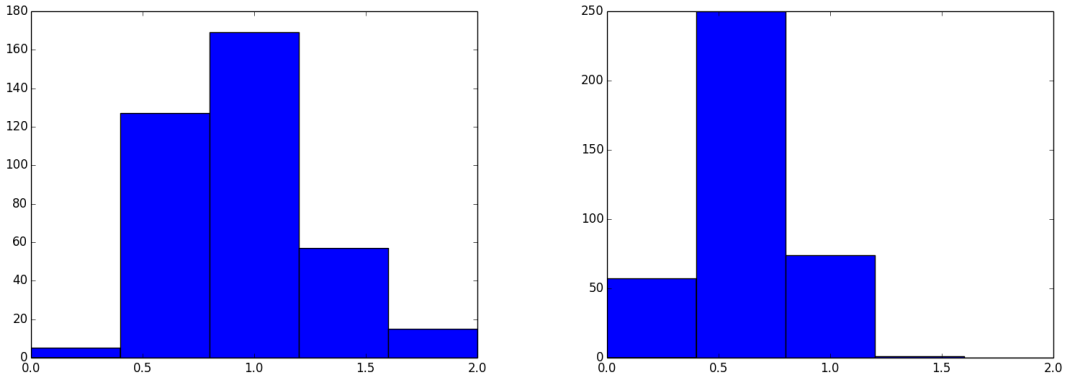
Miller verified these results with the use of the Pooled Two-Sample t-Procedure and the Unpooled Two-Sample t-Procedure, standard statistical tests which check for strength of difference in the means of the two samples. Since in his case the number of degrees of freedom was so large, he was able to use the Central Limit Theorem and replace the t-statistic with a z-statistic. He was then able to obtain strong evidence against the null hypothesis that the two means are equal.

In our experiments, we aimed to verify the results of Miller's experiments when considering a much larger set of L-functions, and L-functions in different families as well. Using computing power at the time, Miller was able to calculate the first normalized zero for log-conductors about 25. He also calculated zeros primarily for certain one-parameter families of elliptic curves. Therefore, he was only able to verify his results for the L-functions of a small subset of curves. Finally, Miller also only analyzed results for curves of rank 0 and 2. In doing so, he did not consider curves of other ranks. Therefore, the set of curves of rank 1 especially is still a significant

subset of elliptic curves which Miller did not analyze.

6.2 Our Results

As described in Chapter 5, we began our experiments by studying families consisting of twists of elliptic curves. In particular, the first family we considered was the twists of the elliptic curve labeled by "11a", which is the first elliptic curve listed in the LMFDB database. In this family, we observed results similar to those of Miller. We first considered the value of the first zero above the central point of all twists of the family of rank 0. We note that for this section, we use the term conductor to refer to the conductor of the associated elliptic curve, but we recall from Chapter 3 that this has a direct connection with the conductor of the L-function itself. We split this set into two subsets of smaller and larger 382 zeros, graphed the values, and performed the same statistical testing as Miller. The graphs of values of the zeros in these two subsets and the statistics can be seen in Figure 6.4.

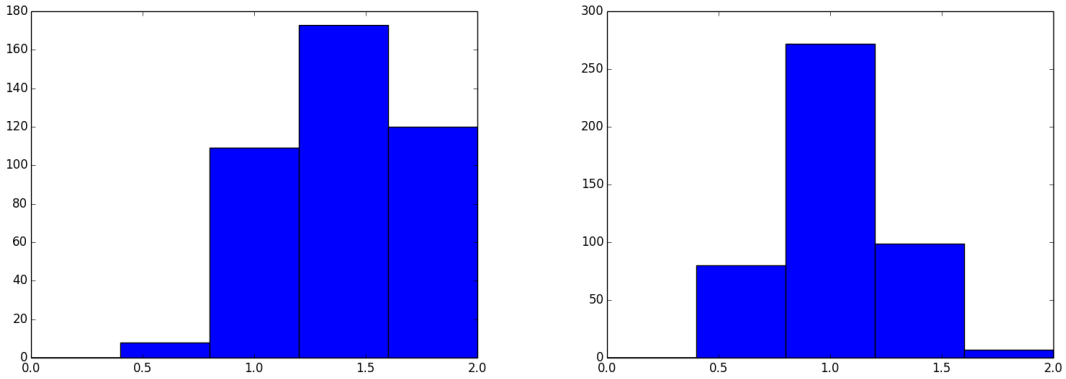


log(conductor)	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
<18.44	0.18	0.21	0.21
>18.44	0.14	0.15	0.08

Figure 6.4: First zero above the central point for rank one L-functions of twists of the elliptic curve 11a with z-test value = 4.9254

As we can see from the graphs, the mean of the zero values appears to be decreasing as the conductor size tends towards infinity, and our statistics verify this observation.

We then considered the value of the first zero above the central point of all twists of "11a" of rank > 0 (we recall that by the BSD conjecture and the parity conjecture that this should be approximately all twists of rank 1). We again split this set into two subsets of 459 zeros each based on conductor size and the graphs of values of the zeros and their relevant statistics can be seen in Figure 6.5.



log(conductor)	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
<18.40	0.49	0.53	0.23
>18.40	0.39	0.39	0.12

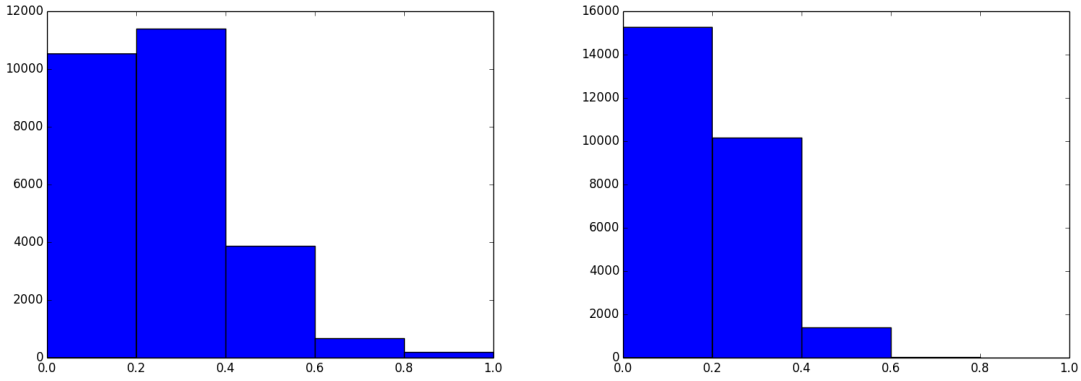
Figure 6.5: First zero above the central point for rank ≥ 1 L-functions of twists of elliptic curve 11a with z-test value = 11.2464

Once again, we observed both graphically and statistically that the mean shifted to the left as the conductor size increased.

The next family we considered was replicating the twisting we did for "11a''", and this time doing it for all elliptic curves up to conductor 230. In this family, we also observed results similar to those of Miller. For the family of rank 0, we found had two subsets of 26837 zeros each, and the results can be seen in Figure 6.6.

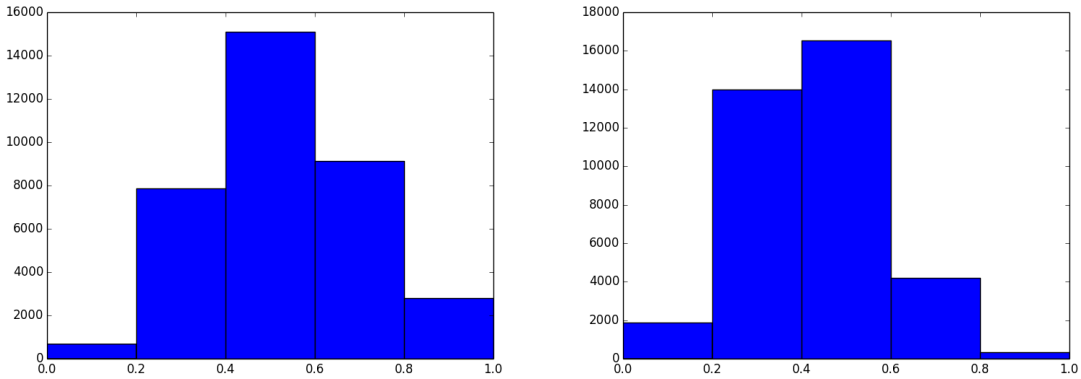
We then considered the value of the first zero above the central point of all twists of rank > 0 (we once again recall that by the BSD conjecture and the parity conjecture that this should be approximately all twists of rank 1). We split this set into two subsets of 36925 zeros each, who results can be seen in Figure 6.7.

In all of these cases seen above, our results were again verifed.



log(conductor)	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
<18.42	0.24	0.27	0.17
>18.42	0.18	0.20	0.11

Figure 6.6: First zero above the central point for rank zero L-functions of all twists of elliptic curves z-test value = 61.0855

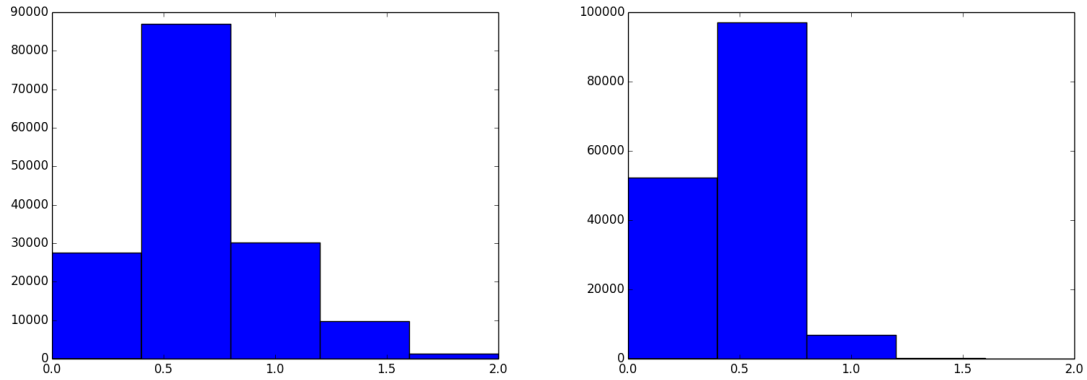


log(conductor)	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
<18.41	0.53	0.56	0.23
>18.41	0.43	0.43	0.15

Figure 6.7: First zero above the central point for rank ≥ 1 L-functions of all twists of elliptic curves z-test value = 90.4681

6.3 Results for All Elliptic Curves

The next family we considered was the family of all the L-functions associated with elliptic curves up to conductor 250000. We note that in this case we split the set into two subsets of 156160 and 156156 zeros respectively based on conductor size (in this case, conductor size is regarded directly, since log(conductor) is too small of a number for most of these elliptic curves). Results for the family of rank 0 can be seen in Figure 6.8.



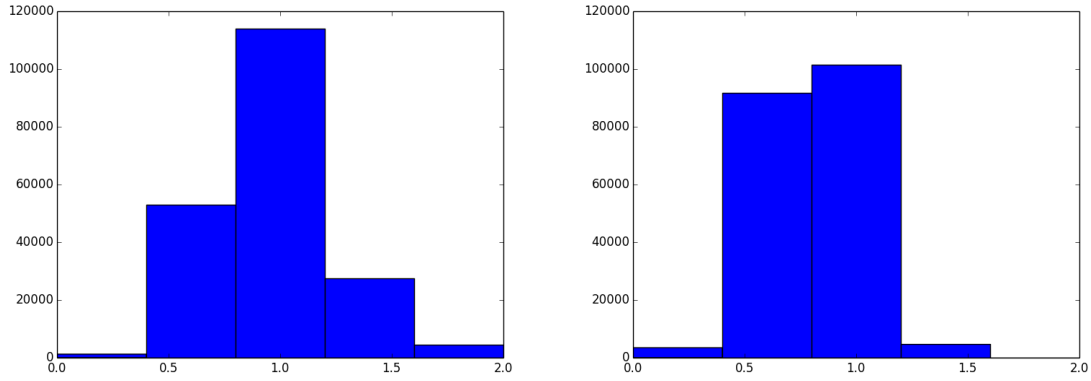
Conductor	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
<120225	0.61	0.67	0.31
>120225	0.48	0.49	0.17

Figure 6.8: First zero above the central point for rank one L-functions of all Elliptic Curves z-test value = 202.1637

We next considered the value of the first zero above the central point of all twists of the family of rank 1. We split this set into two subsets of 201623 and 201261 zeros respectively. Results can be seen in Figure 6.9.

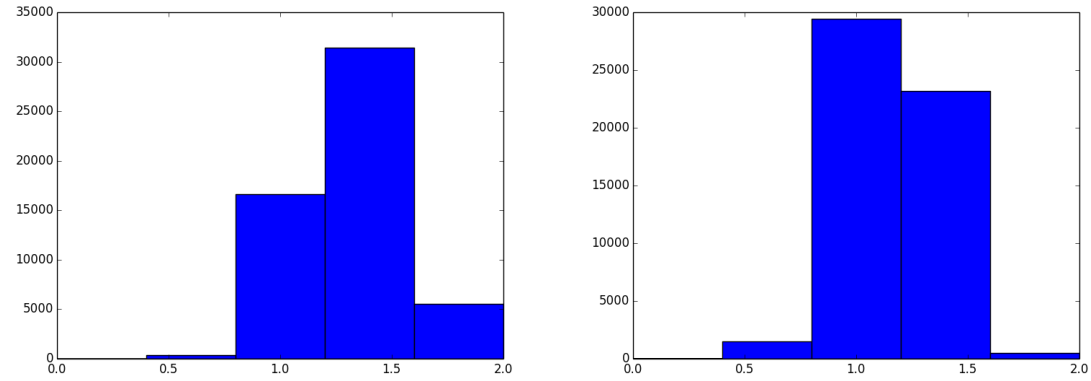
For this type of family, we finally considered the value of the first zero above the central point of all twists of the family of rank ≥ 2, and we note that we can say that almost all of these are actually of rank 2, so we will call them this. We split this set into two subsets of 54647 zeros each, and results can be seen in Figure 6.10.

We note that in all three of these cases, our predictions held. Also, Miller's prediction about the attraction decreasing as the rank of the families increases is also



Conductor	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
<133848	0.94	.97	0.27
>133848	0.81	0.81	0.20

Figure 6.9: First zero above the central point for rank one L-functions of all Elliptic Curves z-test value = 202.1637



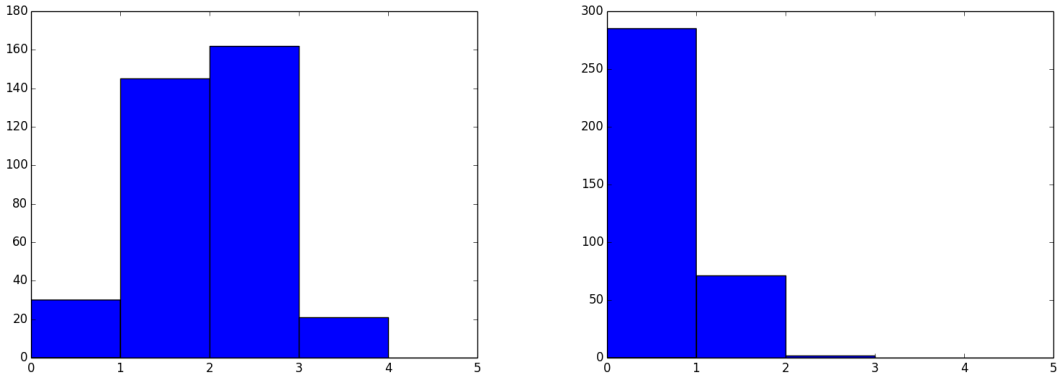
Conductor	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
<139110	1.31	1.32	0.24
>139110	1.17	1.17	0.19

Figure 6.10: First zero above the central point for rank two L-functions of all Elliptic Curves Forms z-test value = 117.8212

evident by comparisons of the graphs and mean values of these three cases.

6.4 Results for Modular Forms of Level 1

Next we continued our experiments by studying the family of L-functions associated with modular forms of level 1 and varying the weight k. In our experiments, we considered $2 \leq k \leq 200$. We split this set into two subsets of 361 and 357 zeros respectively, and considered the family of rank 0. Our results can be seen in Figure 6.11.

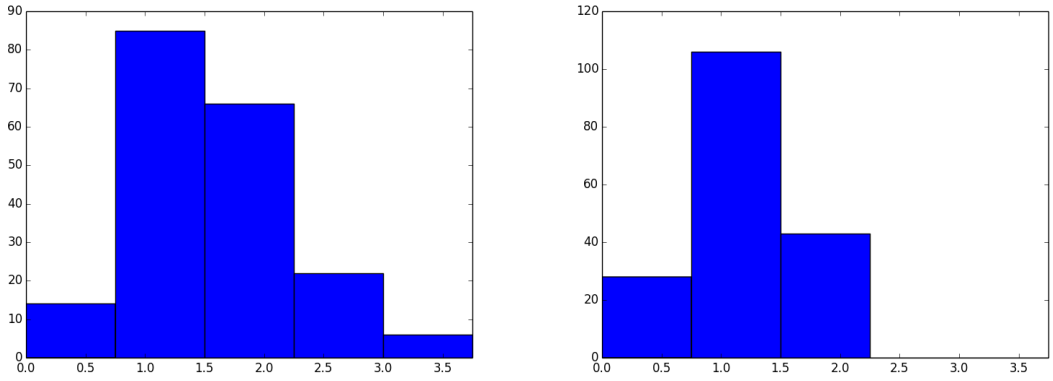


Weight k	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
$\leq$60	2.03	2.03	0.81
$>$60	0.58	0.66	0.44

Figure 6.11: First zero above the central point for rank zero L-functions of Level 1 Modular Forms z-test value = 28.1352

We then considered the value of the first zero above the central point of the family of rank 1. We split this set into two subsets of 198 and 178 zeros respectively, whose results can be seen in Figure 6.12.

We note that in the case of L-functions associated with modular forms of level 1 for the range of weights we investigated, there were no L-functions of rank > 1.



Weight	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
$\leq$146	1.49	1.66	0.88
>146	1.12	1.18	0.42

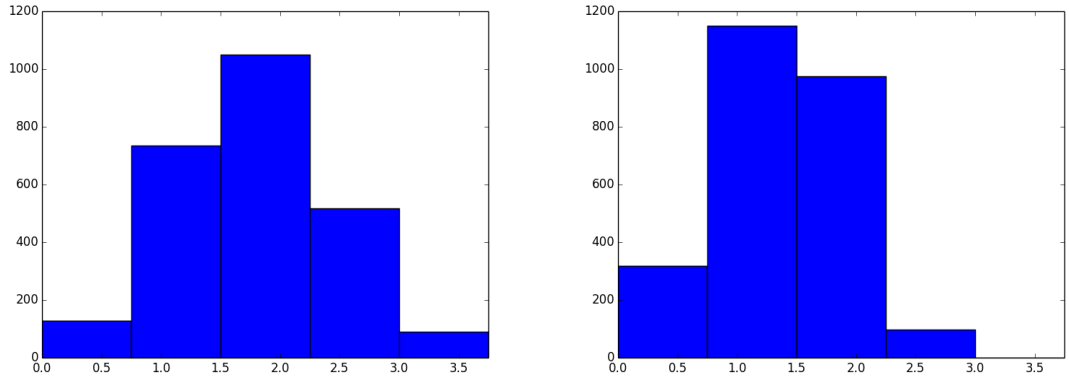
Figure 6.12: First zero above the central point for rank one L-functions of Level 1 Modular Forms z-test value = 6.8646

6.5 Results for Modular Forms of Level 2

Next we continued our experiments by studying the family of L-functions associated with modular forms of varying level N and weight 2. In our experiments, we considered $1 \leq N \leq 700$. We split this set into two subsets of 2540 and 2540 zeros respectively, and considered the family of rank 0. The graphs of values of the zeros in these two subsets can be seen in Figure 6.13.

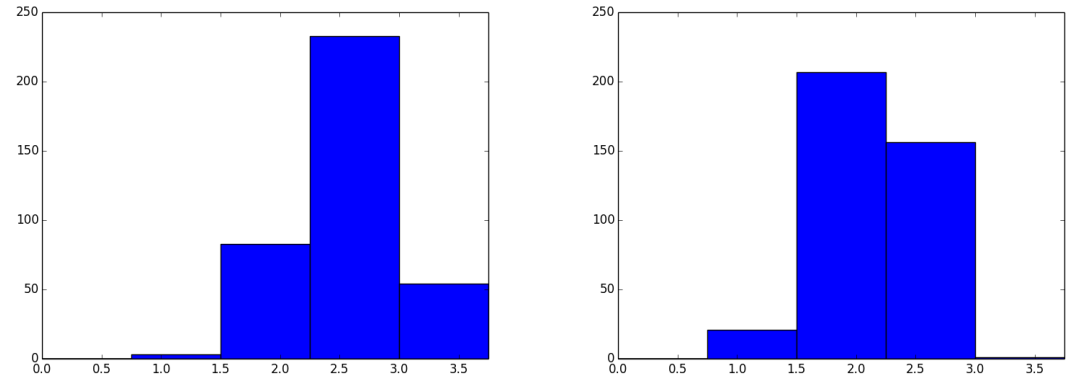
We then considered the value of the first zero above the central point of the family of rank 1. We split this set into two subsets of 386 zeros respectively, whose results can be seen in Figure 6.14.

We note that in the case of L-functions associated with modular forms of level 1 for the range of weights we investigated, there were only 9 L-functions of rank > 1, and thus we will not analyze them.



Level N	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
$\leq$461	1.78	1.80	0.69
>461	1.39	1.39	0.52

Figure 6.13: First zero above the central point for rank zero L-functions of Weight 2 Modular Forms z-test value = 30.4711



Level N	Median $\tilde{\mu}$	Mean μ	StDev σ_μ
$\leq$469	2.55	2.61	0.53
>469	2.16	2.14	0.37

Figure 6.14: First zero above the central point for rank one L-functions of Weight 2 Modular Forms z-test value = 14.3892

6.6 Discussion of Results

In the above sections, we see that in every case of experiments we considered, there existed a statistically significant difference between the means of the lowest zeros of L-functions of lower conductor and those of higher conductor. Specifically, in every case the zeros of lower conductor were significantly larger than those of higher conductor. This indicates that the hypothesis tested by Miller in his work in 2005 in fact appears to be for much larger families of L-functions of elliptic curves, and even for families of modular forms as well. In particular, we found that for all of the families of L-functions associated with elliptic curves and modular forms with finite N that we gathered data for, we observe less attraction of the distribution of the first zero above the central point $s = \frac{1}{2}$. As N increases, the attraction increases, and seems to agree with the random matrix theory model. Also, the attraction is greater for comparable families of L-functions with lesser rank. These experimental results parallel the random matrix theory predictions associated with the distribution of the first zero of a family of L-functions as $N \to \infty$.

6.7 Conclusions and Future Work

In our experiments, we came very close to hitting the upper bounds of current computational limits for L-functions associated with both elliptic curves and modular forms. For example, it would be fairly trivial to extend the the family of L-functions associated with elliptic curves from conductor 250000 to conductor 350000. However, 350000 is a current upper bound for this family, since the Cremona database ends at that value, and no other database of all elliptic curves based on conductor N currently exists. Also, in our work we also came very close to upper bounds for L-functions associated with modular forms. In fact, for the two families we considered, we did hit upper bounds for current computational power for the two families we considered. Moreover, since computations very quickly become difficult for modular forms in Sage, any other families with higher level N or weight k would quickly result in hitting upper bounds as well. Therefore, our experiments actually covered almost all of the computationally possible L-functions assoicated with modular forms.

Due to these computational upper bounds, future work based on the same computational structure that we used would quickly become impossible based on the computing power available today. However, we now have amassed a large amount of

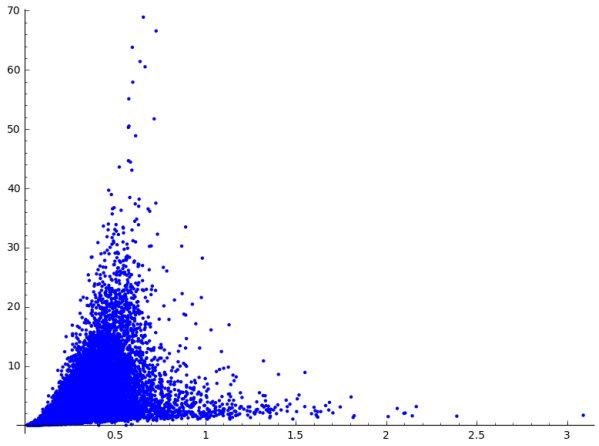


Figure 6.15: First zero value versus value at $L(1/2)$

L-function data files that could be accessed to perform an analysis on any number of other statistics based on L-functions. For example, at the end of our work we also considered a new statistic in our experiments by asking the question: is there any correlation between the value of an L-function at $s = \frac{1}{2}$ and the value of the first zero of the L-function on the critical line. We plot the results of this graph below, with the zero value found on the x-axis and the value of $L(\frac{1}{2})$ found on the y-axis, using the data set of the first $50,000$ rank 0 L-functions in our twists of elliptic curves data set.

In this graph, we see an interesting relationship between the two values which merits further exploration. Analysis of these results and their context within the greater random matrix theory connection could prove to be a very interesting project to pursue.

References

[1]

[2] L-calc code respository. http://code.google.com/p/l-calc/. Accessed as recently as: 3/29/15.

[3] Lmfdb. www.lmfdb.org. Accessed as recently as: 3/29/15.

[4] "primes". "http://science.larouchepac.com/gauss/ceres/InterimII/Arithmetic/Primes/Primes. "Accessed as recently as: 4/22/15".

[5] Sagemath. http://www.sagemath.org. Accessed as recently as: 3/29/15.

[6] Andrew R. Booker. Artin's conjecture, Turing's method, and the Riemann hypothesis. *Experiment. Math.*, 15(4):385–407, 2006.

[7] Christophe Breuil, Brian Conrad, Fred Diamond, and Richard Taylor. On the modularity of elliptic curves over $\mathbf{Q}$: wild 3-adic exercises. *J. Amer. Math. Soc.*, 14(4):843–939 (electronic), 2001.

[8] Daniel Bump. The Rankin-Selberg method: an introduction and survey. In *Automorphic representations, L-functions and applications: progress and prospects*, volume 11 of *Ohio State Univ. Math. Res. Inst. Publ.*, pages 41–73. de Gruyter, Berlin, 2005.

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

[10] Benedict H. Gross. The arithmetic of elliptic curves—an update. *Arab. J. Sci. Eng. ASJE. Math.*, 34(1D):95–103, 2009.

[11] Nicholas M. Katz and Peter Sarnak. Zeroes of zeta functions and symmetry. *Bull. Amer. Math. Soc. (N.S.)*, 36(1):1–26, 1999.

[12] Neal Koblitz. *Introduction to elliptic curves and modular forms*, volume 97 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1984.

[13] Francesco Mezzadri. How to generate random matrices from the classical compact groups. *Notices Amer. Math. Soc.*, 54(5):592–604, 2007.

[14] Steven J. Miller. Investigations of zeros near the central point of elliptic curve L-functions. *Experiment. Math.*, 15(3):257–279, 2006. With an appendix by Eduardo Dueñez.

[15] Steven J. Miller and Ramin Takloo-Bighash. *An invitation to modern number theory*. Princeton University Press, Princeton, NJ, 2006. With a foreword by Peter Sarnak.

[16] D. J. Newman. Simple analytic proof of the prime number theorem. *Amer. Math. Monthly*, 87(9):693–696, 1980.

[17] Bernhard Riemann. *Gesammelte mathematische Werke, wissenschaftlicher Nachlass und Nachträge*. Springer-Verlag, Berlin; BSB B. G. Teubner Verlagsgesellschaft, Leipzig, 1990. Based on the edition by Heinrich Weber and Richard Dedekind, Edited and with a preface by Raghavan Narasimhan.

[18] Christoph Schmitt. Calculation of l-functions associated with newforms: Implementation, choice of parameters, and verification of zeros. Master's thesis, Julius-Maximilians Universitaet Wuerzburg, 2010.

[19] Goro Shimura. On the periods of modular forms. *Math. Ann.*, 229(3):211–221, 1977.

[20] 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.

[21] Richard Taylor and Andrew Wiles. Ring-theoretic properties of certain Hecke algebras. *Ann. of Math. (2)*, 141(3):553–572, 1995.

[22] Kelly Devine Thomas. From prime numbers to nuclear physics and beyond. https://www.ias.edu/about/publications/ias-letter/articles/2013-spring/primes-random-matrices, 2013. Accessed as recently as: 3/29/15.

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

[24] Andrew Wiles. The Birch and Swinnerton-Dyer conjecture. In *The millennium prize problems*, pages 31–41. Clay Math. Inst., Cambridge, MA, 2006.

Appendices

Appendix 1

In this appendix, we find the main code we wrote to compute the lcalc files containing the data about an L-function that is necessary to perform computations to learn more about the L-function. The first function allows us to write all of the parameters associated with an L-function into a correctly formated l-calc file.

```
def write_lcalc_file(base_curve, N, coefficient_type, L_function_type,
    number_coefficients, periodic_coefficients, number_gamma_factors,
    first_gamma, first_lambda, Q, omega, poles, coefficient_string, k =
    None, j = None):
      '''Takes in the parameters for an L-function and writes a l-calc
         formatted files with those parameters'''
      #deals with elliptic curves case
      if k == None:
           o = open('EllipticCurves/'+ base_curve + '/'+ str(N) + '.lcalc
               ', 'w')
      #deals with modular forms case
      else:
           o = open('ModularForms/' + 'level_' + str(N) + '_weight_' +
              str(k) + '_' + str(j) +  '.lcalc', 'w')
      o.write(str(coefficient_type) + '\n')
      o.write(str(L_function_type) + '\n')
      o.write(str(number_coefficients) + '\n')
      o.write(str(periodic_coefficients) + '\n')
      o.write(str(number_gamma_factors) + '\n')
      o.write(str(first_gamma) + '\n')
      o.write(str(first_lambda) + '\n')
      o.write(str(Q) + '\n')
      o.write(str(omega) + '\n')
      o.write(str(poles) + '\n')
      o.write(coefficient_string)
      o.close()
```

Next, we see a couple of functions which allows us to take in an elliptic curve object from sage, and find the characteristics of its associated L-function.

```
def coefficient_loop(coefficient_list, base = False):
    '''Creates a list of normalized coefficients of an L-function'''
    long_list = ''
  #if coefficients still need to be normalized
    if base == True:
       for i in range(1, len(coefficient_list)):
            long_list += str(coefficient_list[i]/sqrt(i).n()) + '\n'
   #coefficients are already normalized
   else:
        for i in range(1, len(coefficient_list)):
            long_list += str(coefficient_list[i]) + '\n'
   return long_list

def make_base_lcalc_file(first_curve, number_coefficients):
   '''Takes in the LMFDB marker for an elliptic curve and returns its L
      -function parameters, up to a certain number of coeffients'''
   first_curve = first_curve
   #creates elliptic curve object in sage
   E = EllipticCurve(first_curve)
   N = E.conductor()
   #sign of the functional equation
   omega = str(E.root_number()) + ' 0'
   number_coefficients = number_coefficients
   #the following few lines are l-calc constants
   coefficient_type = 2
   L_function_type = 2
   periodic_coefficients = 0
   number_gamma_factors = 1
   first_gamma = 1
   first_lambda = ".5 0"
   #conductor of the L-function
   Q = sqrt(N).n()/(2*pi.n())
   poles = 0
   coefficient_list = E.anlist(number_coefficients)
   coefficient_string = coefficient_loop(coefficient_list, True)
   #once parameters are gathered, sends them to write_lcalc_file to be
      made into a file
   write_lcalc_file(first_curve, N, coefficient_type, L_function_type,
      number_coefficients, periodic_coefficients, number_gamma_factors
      , first_gamma, first_lambda, Q, omega, poles, coefficient_string
      )
```

Here, we see the same sort of function, but this time it is modified to work for modular forms instead of elliptic curves.

```
def gen_make_lcalc_file(newform, number_coefficients, real_embedding=
    None, j = None):
    '''Takes in the Sage newform object and if necessary a specific real
        embedding of that object, and returns the parameters for its
       associated L-function, up to the number of coefficients'''
    n = newform
    k = n.weight()
    N = n.level()
    #sign of the functional equation
    eps = (-1)^(k/2)*n.atkin_lehner_eigenvalue()
    #the following few lines are l-calc constants
    coefficient_type = 2
    L_function_type = 2
    periodic_coefficients = 0
    number_gamma_factors = 1
    first_gamma = 1
    first_lambda = str((k-1)/2.n(100)) + " 0"
    #conductor of the L-function
    Q = sqrt(N).n()/(2*pi.n())
    poles = 0
    omega = str(eps) + " 0"
    coefficient_string = ''
    if real_embedding is None:
       for i in range(1, number_coefficients+1):
           val = (n[i]/i^((k-1)/2)).n(100)
     coefficient_string += str(val) + '\n'
    else:
       for i in range(1, number_coefficients + 1):
           val = (real_embedding(n[i])/i^((k-1)/2)).n(100)
     coefficient_string += str(val) + '\n'
    #once parameters have been gathered, writes them into an l-calc file
    write_lcalc_file(newform, N, coefficient_type, L_function_type,
       number_coefficients, periodic_coefficients, number_gamma_factors
       , first_gamma, first_lambda, Q, omega, poles, coefficient_string
       , k, j)
```

Here, we see a function which allows us to take in an l-calc file for our base L-function, and create a series of l-calc files by twisting the data in the base one.

```
def twist_base_lcalc_file(path_to_base_file):
    '''Takes in a base L-function file and creates the files for a
       family of twists'''
    lines = [line.strip() for line in open(path_to_base_file)]
```

```
first_curve = os.path.dirname(path_to_base_file)
filename = os.path.basename(path_to_base_file)
base_N = int(os.path.splitext(filename)[0])
#range can be adjusted to include more twists
for i in range(0,-8000, -1):
  f = is_fundamental_discriminant(i)
  if f:
       d = i
       N = base_N*d^2
       #all of these coefficient numbers were found through
           experimentation
       if N < 300000:
              number_coefficients = 10000
       elif N < 1700000:
              number_coefficients = 20000
       elif N < 5900000:
              number_coefficients = 40000
       elif N < 35000000:
              number_coefficients = 80000
       elif N < 144000000:
              number_coefficients = 160000
       elif N < 400000000:
              number_coefficients = 320000
       else:
              break
       coefficient_type = lines[0]
       L_function_type = lines[1]
       periodic_coefficients = lines[3]
       number_gamma_factors = lines[4]
       first_gamma = lines[5]
       first_lambda = lines[6]
       #conductor of the new L-function
       Q = sqrt(N).n()/(2*pi.n())
       #sign of the functional equation for the new L-function
       omega = str(int(lines[8][0:2])*d/abs(d)*kronecker_symbol(d,
           base_N)) + ' 0'
       poles = lines[9]
       n = 1
       length = len(lines)
       coefficient_list = [0]
       #adjusting coefficients for the new L-function
       for x in range(10, number_coefficients + 9):
            a_n = float(lines[x])
            coefficient_list += [a_n*kronecker_symbol(d,n)]
            n += 1
       coefficient_string = coefficient_loop(coefficient_list)
       write_lcalc_file(first_curve, N, coefficient_type,
           L_function_type, number_coefficients,
```

```
            periodic_coefficients, number_gamma_factors, first_gamma,
             first_lambda, Q, omega, poles, coefficient_string)
```

Here, we see a function which allows us to loop through and extract all the elliptic curves from the Cremona database.

```
def loop_through_database(first_curve, last_curve, number_coefficients =
    10000):
    '''Takes in a range and finds all the elliptic curves within that
       condutor range in the Cremona Database'''
    C = CremonaDatabase()
    current_curve = first_curve
    while int(current_curve) <= int(last_curve):
            A = C.allcurves(current_curve)
            if A != {}:
               keys = []
               for key in A.iteritems():
                     #print(key[0][0])
                     if key[0][0] not in keys:
                     keys += [key[0][0]]
              for i in range(len(keys)):
                     curve_name = current_curve + keys[i]
                     #print(current_curve + keys[i])
                     E = EllipticCurve(curve_name)
                     N = E.conductor()
                     path_to_base_file = 'EllipticCurves/' + curve_name
                         + '/' + str(N) + '.lcalc'
                     current_curve_directory = 'EllipticCurves/' + str(
                        curve_name)
                     make_base_lcalc_file(curve_name,
                        number_coefficients)
                     run_lcalc_files(current_curve_directory)
             current_curve = str(int(current_curve) + 1)
```

Finally, here we have a function which allows us to loop through all the modular forms within a certain level and/or weight range.

```
def make_all(Nmin, Nmax, kmin, kmax, number_coefficients = 10000):
    '''Takes in a range of levels and range of weights and creates L-
       functions of all modular forms within those ranges'''
    for k in range(kmin, kmax, 2):
         for N in range(Nmin, Nmax, 1):
              print k, N
              #creates newforms object
              SkN = Newforms(N, k, names='a')
```

```
            dimS = len(SkN)
            for i in range(dimS):
                f = SkN[i]
                K = f.hecke_eigenvalue_field()
                #checks if the field is rational
                if K == QQ:
                    gen_make_lcalc_file(f, number_coefficients)
                #if the field is not rational, have to find the real
                    embeddings
                else:
                    embs = f.hecke_eigenvalue_file():
                        real_embeddings()
                    degK = len(embs)
                    for j in range(degK):
                        gen_make_lcalc_file(f,
                            number_coefficients, embs[j], j)

    return
```

Appendix 2

In this appendix, we find the code which allows us to create shell files which can be run on a computing cluster.

```
def run_lcalc_files(directory):
    for root, _, files in os.walk(directory):
        for file in files:
            filename = os.path.splitext(file)[0]
            ext = os.path.splitext(file)[1]
            if ext[0:2] == '.l':
                newfile1 = directory + '/' + filename + '.1.txt'
                f = open(newfile1, 'w')
                f.write('#!/bin/bash \n')
                f.write('#PBS -N ' + filename + ' \n')
                f.write('#PBS -o /home/accounts/student/k/keh021/
                    Honors_Thesis/' + directory + ' \n')
                f.write('#PBS -e /home/accounts/student/k/keh021/
                    Honors_Thesis/' + directory + ' \n')
                f.write('#PBS -M keh021@bucknell.edu \n')
                f.write('cd "$PBS_O_WORKDIR" \n')
                f.write('./lcalc -v -F ' + '~/Honors_Thesis/' +
                    directory + '/' + filename + '.lcalc' + ' > ' +
                      '~/Honors_Thesis/' + directory + '/' +
                    filename + '.dat' + ' \n')
                f.write('exit 0')
                f.close()
                newfile2 = directory + '/' + filename + '.2.txt'
                o = open(newfile2, 'w')
                o.write('#!/bin/bash \n')
                o.write('#PBS -N ' + filename + ' \n')
                o.write('#PBS -o /home/accounts/student/k/keh021/
                    Honors_Thesis/' + directory + ' \n')
                o.write('#PBS -e /home/accounts/student/k/keh021/
                    Honors_Thesis/' + directory + ' \n')
```

```
o.write('#PBS -M keh021@bucknell.edu \n')
o.write('cd "$PBS_O_WORKDIR" \n')
o.write('./lcalc -z 5 -F ' + '~/Honors_Thesis/' +
    directory + '/' + filename + '.lcalc' + ' >> '
    + '~/Honors_Thesis/' + directory+ '/'+
    filename + '.dat' + ' \n')
o.write('exit 0')
o.close()
```

Appendix 3

In this appendix, we find the code used to do the statistical analysis on the calculated zeros of L-functions. These first few functions show how we collect the zeros from different types of L-functions into files based on the rank of the L-function.

```
def find_rank_lists(directory):
    '''Takes in a directory of twists of L-functions of elliptic curves
       and returns two files composed of lines of: conductor, first
       zero value, L(1/2) value,where one of the files consists of
       this data for L-functions of rank
zero, and one consists of the data for L-functions of higher rank'''
    for root, dirs, files in os.walk(directory):
        for dir in dirs:
            f = open(directory + '/' + dir + '/rank_0_zeros.txt', 'w
                ')
            g = open(directory + '/' + dir + '/other_rank_zeros.txt'
                , 'w')
            for root, dirs, files in os.walk(directory + '/' + dir):
                for file in files:
                    #conductor of elliptic curve associated with L-
                        function
                    filename = os.path.splitext(file)[0]
                    path_to_file = str(directory) + '/' + str(dir)
                        + '/' + str(file)
                    lines = [line.strip() for line in open(
                        path_to_file)]
                    try:
                        #y-value of first zero of L-function
                        zero = lines[1]
                        #the y-value of the first zero is also
                            zero, move to the second zero
                        if float(zero) <= .000001:
                            zero = lines[2]
                            #Write data into rank 0 file
```

```
                                        g.write(filename + ',' + zero + ','
                                            + lines[0] + '\n')
                                    else:
                                        #Write data into other rank zeros
                                            file
                                        f.write(filename + ',' + zero + ','
                                            + lines[0] + '\n')
                                except:
                                    pass
        f.close()
        g.close()

def find_rank_lists_ecs(directory):
        '''Takes in a directory of L-functions of elliptic curves and
            returns three files composed of lines of: conductor, first zero
             value, L(1/2) value, where one of the files consists of this
            data for L-functions of rank zero, one consists of the data for
             L-functions of rank one, and one consists of the data for L-
            functions of higher rank'''
        f = open(directory + '/rank_0_zeros.txt', 'w')
        g = open(directory + '/rank_1_zeros.txt', 'w')
        h = open(directory + '/higher_rank_zeros.txt', 'w')
        for root, dirs, files in os.walk(directory):
            for dir in dirs:
                for root, dirs, files in os.walk(directory + '/' + dir)
                    :
                    for file in files:
                        #conductor of elliiptic curve associated with
                            L-functions
                        filename = os.path.splitext(file)[0]
                        ext = os.path.splitext(file)[1]
                        path_to_file = str(directory) + '/' + str(dir)
                            + '/' + str(file)
                        path_to_lcalc_file = str(directory) + '/' +
                            str(dir) + '/' + str(filename) + '.lcalc'
                        if ext == '.dat':
                            lines = [line.strip() for line in open(
                                path_to_file)]
                            lines2 = [line.strip() for line in open(
                                path_to_lcalc_file)]
                            #omega of L-function, taken directly from
                                .lcalc file for L-function
                            omega = lines2[8]
                            try:
                                #y-value of first value of L-
                                    function
                                zero = lines[1]
```

```
                                    #the y-value of the first zero is
                                        also zero, move to the second
                                        zero
                                    if float(zero) <= .000001:
                                        zero = lines[2]
                                        #if the sign of the functional
                                            equation is negative
                                        if omega[0] == '-':
                                            #write data into rank 1
                                                file
                                            g.write(filename + ',' +
                                                zero + ',' + lines[0]
                                                 + '\n')
                                        #if the sign of the functional
                                            equation is positive
                                        if omega[0] == '1':
                                            #write data into higher
                                                rank file
                                            h.write(filename + ',' +
                                               zero + ',' + lines[0]
                                               + '\n')
                                    else:
                                            #write data into rank 0
                                                file
                                            f.write(filename + ',' +
                                                zero + ',' + lines[0]
                                                 + '\n')
                                except:
                                    pass
        f.close()
        g.close()
        h.close()

def find_rank_lists_modforms(directory):
        '''Takes in a directory of L-functions of modular forms and returns
            three files composed of lines of: conductor, first zero value,
            L(1/2) value, where one of the files consists of this data for
            L-functions of rank zero,one consists of the data for L-
           functions of rank one, and one consists of the data for L-
           functions of higher rank'''
        f = open(directory + '/rank_0_zeros.txt', 'w')
        g = open(directory + '/rank_1_zeros.txt', 'w')
        h = open(directory + '/higher_rank_zeros.txt', 'w')
        for root, dirs, files in os.walk(directory):
            for file in files:
                filename = os.path.splitext(file)[0]
                #if we are fixing the weight and varying the level
                if "Weight" in directory:
```

```
        #level of modular form associated with L-functions
        numb = filename.split('_')[1]
    #if we are fixing the level and varying the weight
     if "Level" in directory:
          try:
              #weight of modular forms associated with L-
                  functions
              numb = filename.split('_')[3]
          except:
              pass
    else:
          pass
    ext = os.path.splitext(file)[1]
    path_to_file = str(directory) + '/' + str(file)
    path_to_lcalc_file = str(directory) + '/' + filename + '.
        lcalc'
    if ext == '.dat':
           lines = [line.strip() for line in open(path_to_file)
               ]
           lines2 = [line.strip() for line in open(
               path_to_lcalc_file)]
          #omega of L-function, taken directly from .lcalc file
               for L-function
          omega = lines2[8]
          try:
              #y-value of first value of L-function
              zero = lines[1]
              #the y-value of the first zero is also zero, move
                   to the second zero
              if float(zero) <= .000001:
                    zero = lines[2]
                   #if the sign of the functional equation is
                       negative
                    if omega[0] == "-":
                         #write data into rank 1 file
                         g.write(numb + ',' + zero + ',' +
                             lines[0] + '\n')
                  #if the sign of the functional equation is
                      positive
                   if omega[0] == "1":
                        #write data into higher rank file
                         h.write(numb + ',' + zero + ',' +
                             lines[0] + '\n')
             else:
                  #write data into rank 0 file
                  f.write(numb + ',' + zero + ',' + lines[0] +
                       '\n')
        except:
```

```
                    pass
    f.close()
    g.close()
    h.close()
```

In the next function, we see how we were able to gather zeros data on from multiple directories and group it together to gather statistics for very large families.

```
def find_giant_rank_list(directory):
    '''Takes in a directory and combines all of the composite zeros
       files from subdirectories, then writes out these complete files
        within the directory'''
    f = open(directory + '/rank_0_zeros.txt', 'w')
    g = open(directory + '/rank_1_zeros.txt', 'w')
    h = open(directory + '/higher_rank_zeros.txt', 'w')
    for root, dirs, file in os.walk(directory):
        for dir in dirs:
            for root, dirs, files in os.walk(directory + '/' + dir):
                for file in files:
                    if file == 'rank_0_zeros.txt':
                        filename = open(directory + '/' + dir + '/' + file, 'r')
                        f.write(filename.read())
                    elif file == 'rank_1_zeros.txt':
                        filename = open(directory + '/' + dir + '/' + file, 'r')
                        g.write(filename.read())
                    elif file == 'higher_rank_zeros.txt':
                        filename = open(directory + '/' + dir + '/' + file, 'r')
                        h.write(filename.read())
                    else:
                        pass
    f.close()
    g.close()
    h.close()
```

Finally, this set of functions shows how we compute the statistics from the composite zeros files based on rank and print them out into files.

```
def mean(lst):
    '''Computes the mean of the values in a list'''
    return float(sum(lst)) / len(lst)

def median(lst):
    '''Computes the median of the values in a list'''
    lstlen = len(lst)
    if lstlen == 0:
        return None
    index= (lstlen - 1) // 2
```

```
    if (lstlen % 2):
        return lst[index]
    else:
        return lst[index]

def standard_deviation(lst):
    '''Computes the standard deviation of the values in a list'''
    avg = mean(lst)
    variance = map(lambda x: (x - avg)^2, lst)
    return sqrt(mean(variance))

def pooled_t_test(lst1, lst2):
    '''Takes in two lists and returns the t-statistic and degrees of
       freedom for a pooled two-sample t-test'''
    n_1 = len(lst1)
    n_2 = len(lst2)
    x_1 = mean(lst1)
    x_2 = mean(lst2)
    s_1 = standard_deviation(lst1)
    s_2 = standard_deviation(lst2)
    s_p = sqrt(((n_1 - 1)*s_1^2 + (n_2 - 1)*s_2^2)/(n_1 + n_2 -2))
    #t-statistic
    t = (x_1 - x_2)/(s_p * sqrt(1/n_1 + 1/n_2))
    #degrees of freedom
    DF = n_1 + n_2 - 2
    list = [t, DF]
    return list

def unpooled_t_test(lst1, lst2):
    '''Takes in two lists and returns the t-statistic and degrees of
       freedom for an unpooled two-sample t-test'''
    n_1 = len(lst1)
    n_2 = len(lst2)
    x_1 = mean(lst1)
    x_2 = mean(lst2)
    s_1 = standard_deviation(lst1)
    s_2 = standard_deviation(lst2)
    #t-statistic
    t = (x_1 - x_2)/(sqrt(s_1^2/n_1 + s_2^2/n_2))
    #degrees of freedom
    DF = (s_1^2/n_1 + s_2^2/n_2)^2/(1/(n_1 - 1)*(s_1^2/n_1)^2 + 1/(n_2
       -1) *(s_2^2/n_2)^2)
    list = [t, DF]
    return list

def find_stats(directory, file):
    '''Takes in a composite file of zeros and returns a stats file
       analyzing the zero data'''
```

```
    h = open(directory + '/' + file + '.stats.txt', 'w')
    lines = [line.strip() for line in open(directory + '/' + file)]
    #parameter being varied in the family of L-functions
    numbs = []
    for line in lines:
        numb, zero, value = line.split(',')
        numbs += [int(numb)]
    middle = median(sorted(numbs))
    if middle == None:
        pass
    else:
        #splits data set into smaller conductor range and larger
            conductor range
        small_zeros = []
        large_zeros = []
        for line in lines:
         numb, zero, value = line.split(',')
                  if int(numb) <= middle:
                      small_zeros += [float(zero)]
                  else:
                      large_zeros += [float(zero)]
        s = sorted(small_zeros)
        l = sorted(large_zeros)
        h.write(str(middle) + '\n')
        h.write(str(len(s)) + '\n')
        h.write(str(len(l)) + '\n')
        h.write(str(mean(s)) + '\n')
        h.write(str(mean(l)) + '\n')
        h.write(str(median(s)) + '\n')
        h.write(str(median(l)) + '\n')
        h.write(str(standard_deviation(s)) + '\n')
        h.write(str(standard_deviation(l)) + '\n')
        h.write(str(pooled_t_test(s,l)) + '\n')
        h.write(str(unpooled_t_test(s,l)) + '\n')
        h.close()

def loop_data(directory):
    '''Takes in a directory and computes the statistics of all three of
        the composite zeros files'''
    for root,dirs,files in os.walk(directory):
        for dir in dirs:
            find_stats(directory + '/' + dir, 'rank_0_zeros.txt')
            find_stats(directory + '/' + dir, 'rank_1_zeros.txt')
            find_stats(directory + '/' + dir, 'higher_rank_zeros.txt'
                )
```