Lead Changes in the Prime Number Race

Jason Sneed

1. Introduction and History

In 1837, Dirichlet proved that for $l, k \in \mathbb{N}$ such that $(l, k) = 1$, there are infinitely many primes, p, of the form $p = kn+l$, i.e., $p \equiv l \pmod{k}$. Chebyshev [Ch] took this idea in a new direction when he compared the primes in two different arithmetic progressions. In 1853, he noted that there were "more" primes congruent to 3 (mod 4) than 1 (mod 4). Since Dirichlet proved that were infinitely many primes of both forms, here we mean that he looked at numbers of form $4n + 1$ and $4n + 3$ that were less than some quantity x and determined that the $4n + 3$ category had more primes for the x values that he tried.

Let A_k be the set of residue classes l modulo k, such that $(a, l) = 1$. Define

$$\pi(x, k, l) = \sum_{\substack{p \leq x \\ p \equiv l \pmod{k}}} 1.$$

The prime number theorem for arithmetic progressions states that

$$\pi(x, k, l) \sim \frac{x}{\varphi(k) \log(x)}, \tag{1}$$

where φ is Euler's function and $l \in A_k$ [Da]. Note that for any $l \in A_k$, we have the same asymptotic formula for $\pi(x, k, l)$. Let

$$\Delta(x, k, l_1, l_2) = \pi(x, k, l_1) - \pi(x, k, l_2),$$

and we assume that $l_1 \not\equiv l_2$. The simplest prime number races involve looking at two arithmetic progressions with the same modulus and considering which residue class, l_1, $l_2 \in A_k$, has more primes up to x (in other words, has the lead in the race). More complicated race problems have also been formulated involving multiple arithmetic progressions and arithmetic progressions with different moduli.

Even with the simple two arithmetic progression race, there are many interesting problems that can be considered such as: What is the x value corresponding to the first lead change in the race (the first sign change of $\Delta(x, k, l_1, l_2)$)? Are there infinitely many lead changes as x tends to infinity (infinitely many sign changes for $\Delta(x, k, l_1, l_2)$)? What proportion of the time is one in the lead?

Some of these questions have already been answered. For Chebyshev's case, $\Delta(x, 4, 3, 1)$ is negative for the first time at $x = 26,861$ (Leech [Le]). $\Delta(x, 3, 2, 1)$ is negative for the first time at $x = 608,981,813,029$ (Bays and Hudson [BH]). The phenomenon that a race almost always has one particular leader or is skewed towards one arithmetic progression is now known as "Chebyshev's Bias". The origin of the bias is explained thoroughly in [RS], [FK], and [GM].

2000 Mathematics Subject Classification: 11N13, 11N05 *Key words and phrases:* distribution of primes, prime number races.

Here we focus on the question: Does the sign of $\Delta(x,k,l_1,l_2)$ change infinitely many times as x tends to infinity? In 1914, Littlewood [Li] showed that $\Delta(x,4,3,1)$ and $\Delta(x,3,2,1)$ each change sign infinitely many times as x tends to infinity, answering the question for the modulo 3 and 4 cases. The goal of this paper will be to describe techniques and tools to answer this question given any modulus k and any two residue classes, l_1, $l_2 \in A_k$.

There are a few general results in this area that allow us to conclude immediately that the sign of $\Delta(x,k,l_1,l_2)$ changes infinitely often for certain races. Let $L(s,\chi)$ denote a Dirichlet L-function with χ a Dirichlet character.

Definition 1. *(Haselgrove's condition for modulus k)* $L(s,\chi) \neq 0$ *for* $0 < s < 1$ *for all* $\chi \bmod k$.

Haselgrove's condition has been shown for $k \leq 72$ [Ru] as well as some larger moduli. It is not difficult to verify Haselgrove's condition for a particular k using Michael Rubinstein's "lcalc" program (which we make use of in section 3). In fact, a slightly weaker statement is required for the results mentioned in this section and our results later (see section 2). We set $N_k(l)$ to be the number of incongruent solutions, u, to the congruence $u^2 \equiv l \pmod{k}$, and note that this will be zero for a non-residue and will be positive (depending only on k) for any quadratic residue.

Theorem A. *(Kátai [Ka]) Assuming Haselgrove's condition for* k *and* $N_k(l_1) = N_k(l_2)$*, then* $\Delta(x,k,l_1,l_2)$ *changes sign infinitely often.*

As a result, we need only study races involving a quadratic residue and a non-residue.

Theorem B. *(Knapowski and Turán [KT]) If Haselgrove's condition is true for k, then for any* $l \not\equiv 1$, $\Delta(x,k,1,l)$ *changes sign infinitely often.*

Note that for moduli 3, 4, 8, 12, and 24, the only quadratic residue is 1, and the two theorems together allow us to conclude that all races for these moduli have infinitely many sign changes.

A recent result of Vorhauer [Vo] shows that there are infinitely many values of x for which -1 will lead any race assuming that k is large enough.

Theorem C. *Suppose that* $(l,k) = 1$*, that* $l \not\equiv -1 \pmod{k}$*, and that* $k > k_0$ *where* k_0 *is a suitable absolute constant. If Haselgrove's condition is true for k, then*

$$\limsup_{x\to\infty} \frac{\Delta(x,k,-1,l)}{\frac{\sqrt{x}}{\log x}} > c_1$$

where c_1 *is a suitably small positive constant that does not depend on k.*

One point to note is that if $\Delta(x,k,l_1,l_2)$ has infinitely many sign changes for all races modulo k, where k is an odd integer, then the same will hold for the races modulo $2k$ because the prime counting functions are identical.

Many cases involving very small moduli have already been settled. As was mentioned earlier, Littlewood proved that $\Delta(x,3,2,1)$ and $\Delta(x,4,3,1)$ changed sign infinitely often [Li]. Knapowski and Turán [KT] showed the races modulo 8 had infinitely many sign changes and also by Theorem 2 those modulo 12 and 24. Stark was able to show that $\Delta(x,5,4,2)$ has infinitely many sign changes [St].

In 1971, Diamond [Di] defined the concept of N-independence for zeros of L-functions that could be checked to show infinitely many sign changes for a given race. With small values of k and N,

the N-independence condition was reasonable to check, and Grosswald [G2] did this to show that all of the races for moduli 5, 7, 11, 13, 17, and 19 changed sign infinitely often. Earlier Grosswald [G1] had proven that the races for the moduli 43, 67, and 163 had infinitely many sign changes using the fact that for each of these prime moduli there existed a zero ρ with $|\rho| < 1$ for L-functions with real nonprincipal character. These are all of the moduli for which this type of result has been proven that we are aware of at this time. For a more detailed overview of results in this area, the reader is referred to [FK], [GM], and [RS].

The main result of this paper will be the following theorem.

Theorem 1. *All races for moduli $k \leq 100$ have infinitely many lead changes.*

Another result of this work is that Haselgrove's condition has been confirmed for $0 \leq k \leq 100$.

We would like to thank Kevin Ford for our conversations about prime number races and for suggestions that helped improve this project. We also thank Harold Diamond for his feedback on this work.

2. Analytic Tools

Definition 2. *Define*

$$\Psi(x;\chi) = \sum_{n\leq x} \Lambda(n)\chi(n).$$

$$\psi(x,k,l) = \sum_{\substack{n\leq x \\ n \equiv l \pmod k}} \Lambda(n)$$

and the difference $D(x,k,l_1,l_2) = \psi(x,k,l_1) - \psi(x,k,l_2)$. Here $\Lambda(n)$ is the von Mangoldt function.

We begin with the following sum:

$$\sum_{\substack{n\leq x \\ n \equiv l \pmod k}} \frac{\Lambda(n)}{\log n} = \sum_{\substack{p\leq x \\ p \equiv l \pmod k}} \frac{\Lambda(p)}{\log p} + \sum_{\nu=2}^{\infty} \sum_{\substack{p^\nu \leq x \\ p^\nu \equiv l \pmod k}} \frac{\Lambda(p^\nu)}{\log p^\nu}.$$

$$= \pi(x,k,l) + \sum_{\substack{p^2\leq x \\ p^2 \equiv l \pmod k}} \frac{\Lambda(p^2)}{\log p^2} + O(x^{1/3})$$

$$= \pi(x,k,l) + \sum_{\substack{1\leq u\leq k \\ u^2 \equiv l \pmod k}} \frac{\pi(x^{1/2},k,u)}{2} + O(x^{1/3}).$$

By applying (1) to the equality above we obtain

$$\pi(x,k,l) = \sum_{\substack{n\leq x \\ n \equiv l \pmod k}} \frac{\Lambda(n)}{\log n} - \frac{N_k(l)}{\varphi(k)} \frac{x^{1/2}}{\log x} + o\left(\frac{x^{1/2}}{\log x}\right). \tag{2}$$

Let D_k be the set of all the distinct Dirichlet characters modulo k and $C_k = D_k - \{\chi_0\}$ where χ_0 is the principal character. Using partial summation and orthogonality,

$$\varphi(k) \sum_{\substack{n \le x \\ n \equiv l \pmod k}} \frac{\Lambda(n)}{\log n} = \varphi(k) \left(\frac{\psi(x,k,l)}{\log x} + \int_2^x \frac{\psi(t,k,l)}{t \log^2 t} dt \right) \tag{3}$$

$$= \sum_{\chi \in D_k} \overline{\chi(l)} \left(\frac{\Psi(x;\chi)}{\log x} + \int_2^x \frac{\Psi(t;\chi)}{t \log^2 t} dt \right).$$

By (2) and (3),

$$\varphi(k)\Delta(x,k,l_1,l_2) =$$

$$= \sum_{\chi \in C_k} (\overline{\chi}(l_1) - \overline{\chi}(l_2)) \left(\frac{\Psi(x;\chi)}{\log x} + \int_2^x \frac{\Psi(t;\chi)}{t \log^2 t} dt \right) - (N_k(l_1) - N_k(l_2)) \frac{x^{1/2}}{\log x} + o\left(\frac{x^{1/2}}{\log x} \right). \tag{4}$$

Results from [RS] indicate that the first term in the last line of (4) should have a logarithmic mean of zero and oscillations roughly of order $\sqrt{x}/\log x$. In (4), the term, $-(N_k(l_1) - N_k(l_2))\frac{x^{1/2}}{\log x}$, will account for a shift in the mean of $\Delta(x,k,l_1,l_2)$ away from 0 if $N_k(l_1) \neq N_k(l_2)$, which occurs when l_1 or l_2 is a quadratic residue while the other is not. In this case, the non-residue's counting function will be ahead most of the time. If $N_k(l_1) = N_k(l_2)$, one can expect frequent lead changes with each function having the lead about equally often.

Assuming the Generalized Riemann Hypothesis (GRH) and the Grand Simplicity Hypothesis (GSH), Rubinstein and Sarnak [RS] proved that the bias disintegrates as the modulus increases to infinity. Thus, we expect it to be easier to show that there are infinitely many sign changes for a race with large modulus k.

We note that if GRH is false in certain ways, then it follows from Landau's Oscillation Theorem [La] that $\Delta(x,k,l_1,l_2)$ has infinitely many sign changes.

We take

$$\begin{aligned} g(s) = g(s,k,l_1,l_2) &= \varphi(k) \int_1^\infty D(x,k,l_1,l_2) x^{-s-1} dx \\ &= \varphi(k) \int_1^\infty [\psi(x,k,l_1) - \psi(x,k,l_2)] x^{-s-1} dx \end{aligned} \tag{5}$$

By orthogonality of characters,

$$g(s) = \sum_{\chi \in C_k} \left(\overline{\chi(l_1)} - \overline{\chi(l_2)} \right) \int_1^\infty \Psi(x;\chi) x^{-s-1} dx = -\frac{1}{s} \sum_{\chi \in C_k} \left(\overline{\chi(l_1)} - \overline{\chi(l_2)} \right) \frac{L'(s,\chi)}{L(s,\chi)}. \tag{6}$$

Thus, $g(s)$ is analytic for $\Re s > 1$, and the right side of (6) gives a meromorphic continuation of $g(s)$ to the entire complex plane, where the poles (excluding $s = 0$) of $g(s)$ are a subset of the zeros of the Dirichlet L-functions, $L(s,\chi)$. Thus, if we assume that $g(s)$ does not have any real poles for $s > \frac{1}{2}$ but $g(s)$ does have a complex pole, say s_0, such that $\Re s_0 > \frac{1}{2}$, and we take a δ satisfying $\frac{1}{2} < \delta < \Re s_0$, then it is easy to prove by contradiction that

$$\limsup_{x \to \infty} \frac{D(x,k,l_1,l_2)}{x^\delta} = \infty, \quad \liminf_{x \to \infty} \frac{D(x,k,l_1,l_2)}{x^\delta} = -\infty.$$

See [FK] for a more detailed explanation.

The following lemma relates the oscillations of $D(x,k,l_1,l_2)$ with those of $\Delta(x,k,l_1,l_2)$.

Lemma 1. *Let*

$$h(x) = \varphi(k)x^{-1/2}D(x,k,l_1,l_2) - N_k(l_1) + N_k(l_2).$$

If

$$\liminf_{x\to\infty} h(x) < 0 < \limsup_{x\to\infty} h(x),$$

then $\Delta(x,k,l_1,l_2)$ has infinitely many sign changes.

Proof. Straightforward using (4) (see [FK]). □

Since $\delta > \frac{1}{2}$, it follows from the lemma that $\Delta(x,k,l_1,l_2)$ has infinitely many sign changes if GRH is false in such a way that $g(s)$ has a pole at s_0 with $\Re s_0 > \frac{1}{2}$. Note that it is possible for GRH to be false, and $g(s)$ to have no poles with $\Re s > \frac{1}{2}$. This is the case when $L(s,\chi)$ has a zero with $\Re s > \frac{1}{2}$, but $(\overline{\chi(l_1)} - \overline{\chi(l_2)}) = 0$.

Now assume that $g(s)$ has no poles with $\Re s > \frac{1}{2}$. Define $G = \{\gamma_1, \gamma_2, \gamma_3, \ldots\}$ to be the set of strictly positive real numbers such that $\frac{1}{2} + i\gamma_j$ is a pole of $g(s)$ and the corresponding residues

$$a_j = \frac{-1}{\frac{1}{2} + i\gamma_j} \sum_{\chi} m(\frac{1}{2} + i\gamma_j, \chi)[\overline{\chi(l_1)} - \overline{\chi(l_2)}], \tag{7}$$

where $\chi \in C_k$ and $m(\frac{1}{2} + i\gamma, \chi)$ is the order of the pole. Let $\gamma_{-j} = -\gamma_j$ and $a_{-j} = \overline{a_j}$ for $j < 0$. Using the functional equation for $L(s,\chi)$, we note that $g(s)$ will also have a pole at $\frac{1}{2} - \gamma_j i$ with residue $\overline{a_j}$. We define new functions, $A(u)$ and $A_T^*(u)$, where the first is related to the $h(x)$ in Lemma 1 with x replaced by e^u.

Definition 3. *Define*

$$A(u) = \varphi(k)e^{-u/2}D(e^u,k,l_1,l_2) = h(e^u) + N_k(l_1) - N_k(l_2)$$

and for $T > 0$,

$$A_T^*(u) = \sum_{|\gamma_j| \le T} \frac{-w(\gamma_j/T)}{\frac{1}{2} + i\gamma_j} \sum_{\chi} m(\frac{1}{2} + i\gamma_j, \chi)[\overline{\chi(l_1)} - \overline{\chi(l_2)}]e^{i\gamma_j u},$$

where $w(\frac{\gamma_j}{T})$ is a weight function supported on $[-T,T]$ with $w(0) = 1$.

We set $w(x) = (1-|x|)\cos(\pi x) + \pi^{-1}\sin(\pi|x|)$. This weight function comes from a paper by Jurkat and Peyerimhoff [JP], and we explain the reason for this choice in section 4.

Ingham [I] used a variant of $A_T^*(u)$ with the weight, $w(x) = 1-|x|$, arising from the Fejer kernel. More generally, we define what will be called an Admissible Weight Function.

Definition 4. *Let $w(x)$ be called an Admissible Weight Function if it has the following properties:*

1. *$w(x)$ is continuous*
2. *$0 \le w(x) \le 1$ with $w(0) = 1$ and $w(x) = 0$ for $|x| \ge 1$*

Figure 1: 4 vs. 12 mod 29

3. $W(x) = \int_{-\infty}^{\infty} w(t)e^{-2\pi itx}dt$ *is real and positive.*

Note that together these conditions imply that $w(-x) = w(x)$, and $w(x)$ is a convolution square ($w(x) = f * f(x)$ for some function f).

The following lemma is a generalization of Ingham's Theorem [I].

Lemma 2. *Let*

$$F(s) = \int_{-\infty}^{\infty} A(u)e^{-2\pi su}du = \frac{1}{2\pi}\int_{-\infty}^{\infty} A(\frac{\log x}{2\pi})x^{-s-1}dx,$$

such that $A(u)$ *is absolutely integrable on any finite interval,* $0 \leq u \leq U$*, and the integral is convergent in some half plane* $\sigma > \sigma_1 \geq 0$*. Now, define*

$$A^*(u) = \sum_{n=-N}^{N} a_n e^{2\pi i\gamma_n u},$$

where $\gamma_{-n} = -\gamma_n$*,* γ_n *is real, and* $a_{-n} = \overline{a_n}$ *with* a_n *as defined in* (7)*. Let*

$$F^*(s) = \int_0^{\infty} A^*(u)e^{-2\pi su}du = \sum_{n=-N}^{N} \frac{a_n}{2\pi(s - i\gamma_n)}$$

for $\sigma > 0$*. Suppose that* $D(s) = F(s) - F^*(s)$ *is analytic in the region* $\sigma \geq 0$*,* $-T \leq t \leq T$ *for some* $0 < T \leq \gamma_N$*. Then for the Admissible Weight Function* $w(\gamma)$*, we have that*

$$\underline{\lim}\, A(u) \leq \underline{\lim}\, A_T^*(u) \leq \overline{\lim}\, A_T^*(u) \leq \overline{\lim}\, A(u),$$

where

$$A_T^*(u) = \sum_{|\gamma_n| \leq T} w(\frac{\gamma_n}{T})a_n e^{i\gamma_n u}.$$

Proof. Replace $1 - |x|$ with $w(x)$ in Ingham's proof. □

$A_T^*(u)$ is a trigonometric polynomial so it is a uniformly almost periodic function [Be]. Thus, if there exists a u such that

$$A_T^*(u) > N_k(l_1) - N_k(l_2),$$

then

$$\limsup_{u\to\infty} A_T^*(u) > N_k(l_1) - N_k(l_2)$$

and consequently

$$\limsup_{u\to\infty} A(u) \geq \limsup_{u\to\infty} A_T^*(u) > N_k(l_1) - N_k(l_2).$$

Therefore, $\limsup_{u\to\infty} h(e^u) > 0$ since $A(u) = h(e^u) + N_k(l_1) - N_k(l_2)$. Since $A_T^*(u)$ is uniform almost periodic and

$$\Big|\int_0^y A_T^*(u)du\Big| \ll 1$$

Figure 2: 49 vs. 7 mod 60

for all y, then $A_T^*(u)$ must be less than $N_k(l_1) - N_k(l_2)$ infinitely often. Thus, $\liminf_{u\to\infty} h(e^u) < 0$, and by Lemma 2, $\Delta(x, k, l_1, l_2)$ has infinitely many sign changes.

Summarizing the previous discussions, we have the following.

Theorem 2. *Assume Haselgrove's condition for modulus k. If there exist u and T such that $A_T^*(u) > N_k(l_1) - N_k(l_2)$, then $\Delta(x, k, l_1, l_2)$ changes signs infinitely often.*

3. The Sage Program and Results

To check that all of the races for a given modulus have values of u and T such that $A_T^*(u) > N_k(l_1)$, a program was written in Sage. Sage is a Python-based programming language that was created by William Stein (and available on his website: http://modular.math.washington.edu/sage/), and Sage was chosen because it has Michael Rubinstein's "lcalc" program built in. This program computes the imaginary parts of zeros of L-functions. Sage also has other convenient features like the ability to compute tables of Dirichlet character values.

The inputs for the program are the modulus, k, an upper bound for the absolute values of the imaginary parts of the zeros that we're using, T, a range of u values specified by entering: *ustart* and *uend*, and finally a stepsize to move through the range of u values called *step*. $A_T^*(u)$ will be evaluated at the u values : *ustart*, *ustart* + *step*, *ustart* + 2*step*, ..., *uend*. The program then does the following:

1. Arranges relevant residue classes into groups of quadratic residues and non-residues.

2. Obtains imaginary parts of all zeros for all $L(s, \chi)$ with $\chi \in C_k$ and arranges them appropriately.

3. Obtains character values.

4. Runs through all races l_1 vs. l_2 not covered by the theorems of Kátai and Knapowski and Turán. For each case, it checks to see if $A_T^*(u) > N_k(l_1) + 0.05$ within a range of u-values. Some versions of the program also record the maximum value of $A_T^*(u)$ and save the values of u and $A_T^*(u)$ to a file that can be used to make a plot.

The program has been used to evaluate races for each modulus up to 100 that has not been previously explored. The results are listed in the table below. Recall that all moduli of the form $2k$ where k is odd and 3, 4, 5, 7, 8, 11, 12, 13, 17, 19, 24, 43, and 67 will be skipped. Figures 1 and 2 show plots of the $A_T^*(u)$ function for the race of 4 vs. 12 modulo 29 and 49 vs 7 modulo 60, respectively. Calculations of $A_T^*(u)$ were first done using the zeros of L-functions computed by "lcalc". The calculations were then redone using Rumely's files of zeros for L-functions involving characters with modulus up to 72 as well as composite moduli up to 100. Rumely's files did not include the primes 73, 79, 83, 89, and 97 so these cases have only been checked using the Rubinstein zeros. The results were the same regardless of which set of zeros were used, and the computation times using the Rubinstein zeros are listed as well as those for when the Rumely zeros were used. The version of the program that used the Rumely zeros was written in a way that required slightly more time to read in and organize these zeros so the computation times are generally slightly longer. The computation times listed in table tell how long it took to run the calculations on a Sun Ultra 20 machine with a Fedora Core 3 operating system and AMD Opteron 1.0 GHz processor.

Both sets of zeros were calculated using series representations for the L-functions. The series is then truncated and twisted so that the critical line is the real axis. Rumely and Rubinstein did this in different ways, and the reader is referred to [R] and [Ru] for further explanation. Their programs then looked for sign changes of the truncated series and applied the intermediate value theorem to find the location of the zeros. Rumely checked rigorously that no zeros were missed and no "false" zeros were computed. Rumely also computed a lower bound for each zero but no upper bound. Thus, the accuracy is not guarenteed for these zeros or those calculated by "lcalc". However, another Sage program was written to compare the Rubinstein and Rumely zeros used for all races considered in this paper (except for moduli 73, 79, 83, 89, and 97 where only the Rubinstein zeros were available). All zeros agreed to at least 10^{-9}. This agreement gives us confidence that all zeros used are accurate to 10^{-9}.

Appendix A contains a "fast-running" version of the program that reads in imaginary parts of zeros from a file. There are also other versions of the program that will look at an individual race, one that prints the u and $A_T^*(u)$ values to a file that can be used to create a plot as mentioned earlier, one that uses the zeros computed by "lcalc", and one that uses the original weights from Ingham's Theorem. The program listed in Appendix A can be used to quickly extend these results well beyond $k = 100$ or to check any specific modulus k for infinitely many sign changes in all races, but we have stopped at 100 here.

One thing to note is that the difficult cases will be when $N_k(l_1)$ is large because more zeros will generally be required to push $A_T^*(u)$ over the barrier. It is also time consuming when the modulus has many different races to consider, for example when k is a large prime.

modulus (k)	*number of races*	T	$u-range$	*step*	N	*time using Rubinstein zeros (in seconds)*	*time using Rumely zeros (in seconds)*
9	6	50	0-20	0.1	2	0.92	1.30
15	6	150	0-20	0.01	4	84.14	102.28
16	6	60	0-50	0.1	4	3.89	3.31
20	6	60	0-50	0.1	4	7.02	8.57
21	18	150	0-20	0.1	4	27.27	34.27
23	110	60	0-20	0.1	2	48.44	60.52
25	90	60	0-20	0.1	2	33.77	47.10
27	72	50	0-20	0.1	2	20.69	35.28
28	18	60	0-20	0.1	4	8.58	11.46
29	182	50	0-20	0.1	2	108.48	132.65
31	210	40	0-20	0.1	2	119.48	147.68
32	36	120	0-20	0.1	4	34.03	53.49
33	60	50	0-20	0.1	4	28.07	35.32
35	90	50	0-20	0.1	4	44.09	56.30
36	18	50	0-20	0.1	4	4.34	5.76
37	306	30	0-20	0.1	2	164.42	202.34
39	90	30	0-20	0.1	4	24.47	31.11
40	14	300	7-40	0.05	8	154.47	192.71
41	380	30	0-20	0.1	2	213.93	262.35
44	60	150	0-40	0.1	4	129.83	158.19
45	90	80	0-20	0.1	4	81.10	100.10
47	506	30	0-20	0.1	2	351.72	452.07
48	14	80	0-40	0.1	8	24.5	30.78
49	420	30	0-20	0.1	2	228.59	204.11
51	168	150	0-20	0.1	4	399.39	498.07
52	90	150	0-20	0.1	4	197.1	245.74
53	650	30	0-40	0.1	2	631.55	801.01
55	270	100	0-20	0.1	4	548.79	678.17
56	42	100	0-100	0.1	8	71.85	88.56
57	216	120	0-20	0.1	4	494.83	627.32
59	812	30	0-40	0.1	2	818.58	1023.50

In light of Theorem 2, we expect large oscillations from $A_T^*(u)$ for large k. For a fixed value of $N_k(l_1)$, we see from the table that in general the T value required for $A_T^*(u) > N_k(l_1)$ and the CPU time per race decrease as k gets larger.

modulus (k)	*number of races*	T	$u-range$	*step*	N	*time using Rubinstein zeros (in seconds)*	*time using Rumely zeros (in seconds)*
60	14	100	0-100	0.1	8	69.85	84.36
61	870	20	0-100	0.1	2	622.15	771.04
63	216	60	0-50	0.1	2	161.53	215.36
64	168	140	0-20	0.1	4	329.3	400.15
65	396	40	0-100	0.1	4	246.61	322.86
68	168	120	0-20	0.1	4	322.39	402.54
69	330	30	0-100	0.1	4	175.11	216.93
71	1190	15	0-50	0.1	2	623.70	790.25
72	42	80	0-100	0.1	8	43.07	53.80
73	1260	15	0-50	0.1	2	706.52	
75	270	120	0-20	0.1	4	620.53	607.21
76	216	120	0-20	0.1	4	571.59	705.07
77	630	30	0-50	0.1	4	507.64	628.35
79	1482	12	0-100	0.1	2	783.97	
80	82	300	0-20	0.1	8	621.71	773.12
81	702	60	0-20	0.1	2	937.67	1149.60
83	1640	10	0-100	0.1	2	765.99	
84	42	300	0-100	0.1	8	551.55	674.52
85	720	25	0-100	0.1	4	451.08	550.83
87	546	20	0-100	0.1	4	324.93	398.52
88	140	200	0-20	0.1	8	695.44	849.34
89	1892	10	0-100	0.1	2	942.10	
91	918	20	0-100	0.1	4	669.51	807.09
92	330	50	0-20	0.1	4	275.61	338.52
93	630	20	0-100	0.2	4	204.05	246.87
95	918	20	0-100	0.3	4	213.37	266.44
96	84	150	0-20	0.1	8	198.56	244.55
97	2256	10	0-100	0.5	2	315.31	
99	630	20	0-20	0.1	4	448.04	474.00
100	270	40	0-20	0.1	4	155.86	189.56

In cases which we examined, $m(\frac{1}{2}+i\gamma,\chi)=1$ in $A_T^*(u)$. Thus,

$$A_T^*(u)=\sum_{|\gamma|\le T}\frac{-w(\frac{\gamma}{T})}{\frac{1}{2}+i\gamma}\sum_{\chi}[\overline{\chi(l_1)}-\overline{\chi(l_2)}]e^{i\gamma u}=\sum_{\chi}[\overline{\chi(l_1)}-\overline{\chi(l_2)}]\sum_{|\gamma_\chi|\le T}\frac{-w(\frac{\gamma_\chi}{T})}{\frac{1}{2}+i\gamma_\chi}e^{i\gamma_\chi u}$$

Let $A_T'(u)$ be similar to $A_T^*(u)$, with the actual γ values replaced by the calculated values, γ', from the lcalc function or Rumely's file. We bound the difference as

$$|A_T^*(u)-A_T'(u)|=\Big|\sum_{\chi}[\overline{\chi(l_1)}-\overline{\chi(l_2)}]\Big[\sum_{|\gamma_\chi|\le T}\frac{-w(\frac{\gamma_\chi}{T})}{\frac{1}{2}+i\gamma_\chi}e^{i\gamma_\chi u}-\sum_{|\gamma'_\chi|\le T}\frac{-w(\frac{\gamma'_\chi}{T})}{\frac{1}{2}+i\gamma'_\chi}e^{i\gamma'_\chi u}\Big]\Big|$$

$$\le 2\Big|\sum_{\chi}\Big[\sum_{|\gamma_\chi|\le T}\frac{-w(\frac{\gamma_\chi}{T})}{\frac12+i\gamma_\chi}e^{i\gamma_\chi u}-\sum_{|\gamma'_\chi|\le T}\frac{-w(\frac{\gamma'_\chi}{T})}{\frac12+i\gamma'_\chi}e^{i\gamma'_\chi u}\Big]\Big|.$$

Suppose that all $|\gamma'_\chi-\gamma_\chi|\le 10^{-9}$. The zeros of $L(s,\chi)$ with χ a real character come in conjugate pairs, and we pair the corresponding terms together. For complex characters, if γ is the imaginary part of a zero for $L(s,\chi)$, then $-\gamma$ is the imaginary part of a zero for $L(s,\overline{\chi})$. We pair these corresponding terms together as well and have

$$|A_T^*(u)-A'_T(u)|\le 2\Big|\sum_{\chi}\sum_{0\le\gamma_\chi\le T}\Big[\frac{-w(\frac{\gamma_\chi}{T})}{\frac14+{\gamma_\chi}^2}(\cos(\gamma_\chi u)+2\gamma_\chi\sin(\gamma_\chi u))+\frac{w(\frac{\gamma'_\chi}{T})}{\frac14+{\gamma'_\chi}^2}(\cos[\gamma'_\chi u]+2\gamma'_\chi\sin[\gamma'_\chi u])\Big]\Big|$$

$$\le 2\phi(k)\max_{\chi\in C_k}\sum_{0\le\gamma\le T}\Big|\Big[\frac{-w(\frac{\gamma}{T})}{\frac14+\gamma^2}(\cos(\gamma u)+2\gamma\sin(\gamma u))+\frac{w(\frac{\gamma'}{T})}{\frac14+{\gamma'}^2}(\cos[\gamma' u]+2\gamma'\sin[\gamma' u])\Big]\Big|$$

$$\le 2\phi(k)\max_{\chi\in C_k}\sum_{0\le\gamma<10}\Big|\Big[\frac{-w(\frac{\gamma}{T})}{\frac14+\gamma^2}(\cos(\gamma u)+2\gamma\sin(\gamma u))+\frac{w(\frac{\gamma'}{T})}{\frac14+{\gamma'}^2}(\cos[\gamma' u]+2\gamma'\sin[\gamma' u])\Big]\Big|$$

$$+2\phi(k)\max_{\chi\in C_k}\sum_{10\le\gamma\le T}\Big|\Big[\frac{-w(\frac{\gamma}{T})}{\frac14+\gamma^2}(\cos(\gamma u)+2\gamma\sin(\gamma u))+\frac{w(\frac{\gamma'}{T})}{\frac14+{\gamma'}^2}(\cos[\gamma' u]+2\gamma'\sin[\gamma' u])\Big]\Big|.$$

Let $f(t,u)=\frac{-w(\frac{t}{T})}{\frac14+t^2}(\cos(tu)+2t\sin(tu))$. By the Mean Value Theorem,

$$|f(\gamma,u)-f(\gamma',u)|\le|\gamma-\gamma'|\max_t\Big|\frac{\partial}{\partial t}f(t,u)\Big|.$$

All of the computations of $A_T^*(u)$ performed for the results presented here used zeros with imaginary part, $|\gamma|\le 300$, and the range of u was always a subinterval of $[0,100]$. Let $f(t,u)=w(\frac{t}{T})g(t)h(t,u)$ where $|w(\frac{t}{T})|\le 1$ and $|w'(\frac{t}{T})|\le\frac{6}{T}$, $g(t)=\frac{1}{\frac14+t^2}$ with $|g(t)|\le\min(4,\frac{1}{t^2})$ and $|g'(t)|\le\min(5.2,\frac{2}{|t|^3})$, and $h(t,u)=(\cos[tu]+2t\sin[tu])$ with $|h(t,u)|\le 1+2t$ and $\Big|\frac{\partial}{\partial t}h(t,u)\Big|\le|2-u|+2tu\le 98+200t$. For $0\le u\le 100$,

$$\frac{\partial}{\partial t}f(t,u)\le\frac{6}{T}\min(4,\frac{1}{t^2})(1+2t)+\min(5.2,\frac{2}{|t|^3})(1+2t)+\min(4,\frac{1}{t^2})(98+200t).$$

We compute

$$\max_{0\le t<10}\frac{\partial}{\partial t}f(t,u)<951.39,$$

$$\max_{10\le t\le 300}\frac{\partial}{\partial t}f(t,u)<21.148.$$

The number of terms in each sum corresponds to the number of zeros of $L(s,\chi)$ in the critical strip with imaginary part no larger than T, denoted $N(T,\chi)$. For $T\ge 2$, $\frac12 N(T,\chi)\le\frac{T}{2\pi}\log(\frac{kT}{2\pi e})+0.4593\log(kT)+2.756$ (restricting to zeros with positive imaginary part) [Mc]. Thus,

$$|A_T^*(u)-A'_T(u)|\le 2\phi(k)\Big(\frac{10}{2\pi}\log(\frac{10k}{2\pi e})+0.4593\log(10k)+2.756\Big)\times 10^{-9}\times 951.39$$

$$+2\phi(k)\left(\frac{T}{2\pi}\log(\frac{kT}{2\pi e})+0.4593\log(kT)+2.756\right)\times 10^{-9}\times 21.148.$$

Considering the values that were used in the table above, $2\phi(k)(\frac{10}{2\pi}\log(\frac{10k}{2\pi e})+0.4593\log(10k)+2.756)$ is largest when $k=97$, and $2\phi(k)(\frac{T}{2\pi}\log(\frac{kT}{2\pi e})+0.4593\log(kT)+2.756)$ is largest when $k=80$ and $T=300$. Thus,

$$|A_T^*(u)-A_T'(u)|\le 1240.27\times 10^{-9}\times 951.39+21.148\times 10^{-9}\times 22155.44=0.00165$$

Recall that the program actually checked to see if $A_T^*(u)>N_k(l_1)+0.05$. The extra 0.05 was to be safe, and with most races we found u and T so that $A_T^*(u)$ beats this barrier by a substantial amount. In light of the error analysis given above, we conclude that $A_T^*(u)$ does truly beat $N_k(l_1)$ in all cases.

Figure 3: A comparison of the Ingham and Jurkat-Peyerimhoff weight functions

4. Comparison of Weight Functions

We turn back to the question of why we chose the Jurkat-Peyerimhoff weight function over the Ingham weight function. Recall that

$$A_T^*(u)=\sum_{|\gamma_j|\le T}\frac{-w(\frac{\gamma_j}{T})}{\frac{1}{2}+i\gamma}\sum_{\chi}m(\frac{1}{2}+i\gamma,\chi)[\overline{\chi(l_1)}-\overline{\chi(l_2)}]e^{i\gamma_j u},$$

where $w(\frac{\gamma_j}{T})$ stands for our weight function. $A_T^*(u)$ is mainly determined by terms that have small γ_j. Running the program for a particular race, there is generally little difference in the value of $A_T^*(u)$ if 1000 is used instead of 200 for T. Thus, the ideal weight function will be one that is very close to 1 for zeros with the smallest imaginary parts in absolute value. We see in Figure 3 that the Jurkat-Peyerimhoff weight function is much closer to 1 for small values of $\left|\frac{\gamma}{T}\right|$. We do, however, sacrifice some of the contribution of the terms with large zeros.

Jurkat and Peyerimhoff were looking for a weight function that satisfied the following requirements: $w(t)=\int_{-\infty}^{\infty}f(x)f(t-x)dx$, $w(0)=1$, $w(0)=0$, and $|w''(0)|$ is minimal, where f is even, $f(t)=0$ for $|t|\ge\frac{1}{2}$, and $f'\in L_2[-\frac{1}{2},\frac{1}{2}]$. These conditions will result in a weight function that concentrates the most weight near 0. Their weight function was the solution to this problem, and a brief proof is included in [JP]. Figures 4 and 5 show a comparison of some races using the two different weight functions. Many, but not all, of the peaks are higher for the Jurkat-Peyerimhoff weight function.

References

[B] M. Bennet, *Rational approximation toalgebraic numbers of small heights: the diophantine equation* $ax^n-by^n=1$, J. Reine Angew. Math. 535, (2001), 1-49.

Figure 4: 25 vs. 5 mod 42

Figure 5: 7 vs. 5 mod 9

[Be] A. S. Besicovitch, *Almost Periodic Functions*, Dover Publications, New York, 1954.

[BH] C. Bays and R. H. Hudson, *The segmented sieve of Eratosthenes and primes in arithmetic progressions to* 10^{12}, Nordisk Tidskr. Inform. (BIT) **17** (1977), 121-127.

[Ch] P. L. Chebyshev, *Lettre de M. le professeur Tchébyshev á M. Fuss, sur un nouveau théoreme rélatif aux nombres premiers contenus dans la formes* $4n + 1$ *et* $4n + 3$, Bull. de la Classe phys.-math. de l'Acad. Imp. des Sciences St. Petersburg **11** (1853), 208.

[Da] H. Davenport, *Multiplicative Number Theory, 2nd ed., Graduate Texts in Mathematics vol. 74*, Springer-Verlag, New York-Berlin, 1980.

[Di] H. G. Diamond, *Two oscillation theorems*, The Theory of Arithmetic Functions, LNM 251 (Anthony A. Gioia, Donald L. Goldsmith, eds.), Springer-Verlag, 1971, pp. 113-118.

[FH] K. Ford and R. Hudson, *Sign changes in* $\pi_{q,a}(x) - \pi_{q,b}(x)$. Acta Arith. **100** (2001), no. 4, 297–314.

[FK] K. Ford and S. Konyagin, *Chebyshev's conjecture and the prime number race*. IV International Conference "Modern Problems of Number Theory and its Applications": Current Problems, Part II (Russian) (Tula, 2001), 67–91, Mosk. Gos. Univ. im. Lomonosova, Mekh.-Mat. Fak., Moscow, 2002.

[G1] E. Grosswald, *Sur une propriété des racines complexes des fonctions* $L(s, \chi)$, C. R. Acad. Sci. Paris **260** (1965), 4299-4302. (French)

[G2] E. Grosswald, *Oscillation theorems of arithmetical functions*, Trans. Amer. Math. Soc. **126** (1967), 1-28.

[GM] A. Granville and G. Martin, *Prime number races*. Amer. Math. Monthly **113** (2006), no. 1, 1–33.

[I] A. E. Ingham, *On two conjectures in the theory of numbers*, Amer. J. Math. **64** (1942), 313-319.

[JP] W. Jurkat and A. Peyerimhoff, *A constructive approach to Kronecker approximations and its application to the Mertens conjecture*. J. Reine Angew. Math. **286/287** (1976), 322–340.

[Ka] I. Kátai, *Eine Bernerkung zur "Comparative Prime-Number Theory I-VIII" von S. Knapowski und P. Turán*, Ann. Univ. Sci. Budapest. Eötvös Sect. Math. **7** (1964), 33-40. (German)

[KT] S. Knapowski and P. Turán, *Comparative Prime Number Theory I.*, Acta. Math. Sci. Hungar. **13** (1962), 315-342.

[La] E. Landau, *Über einen Satz von Tschebyscheff*, Math. Annalen **61** (1905), 527-550. (German)

[Le] J. Leech, *Note on the distribution of prime numbers*, J. London Math. Soc. **32** (1957), 56-58.

[Li] J. E. Littlewood, *Sur la distribution des nombres premiers*, C. R. Acad. des Sciences Paris **158** (1914), 1869-1872.

[Mc] K. S. McCurley, *Explicit Estimates for the Error Term in the Prime Number Theorem for Arithmetic Progressions*, Math. Comp., Vol. 42,**165** (1984), 265-285.

[R] M. Rubinstein, *Computational methods and experiments in analytic number theory: Recent perspectives in random matrix theory and number theory*, 425-506, London Math. Soc. Lecture Note Ser., 322, Cambridge Univ. Press, Cambridge, 2005.

[RS] M. Rubinstein and P. Sarnak, *Chebyshev's Bias*, J. Exper. Math. **3** (1994), 173-197.

[Ru] R. Rumely, *Numerical computations concerning the ERH*, Math. Comp. 61 (1993), 415-440.

[St] H. M. Stark, *A problem in comparative prime number theory*, Acta Arith. **18** (1971), 311-320.

[Vo] U. M. A. Vorhauer, *A Note on Comparitive Prime Number Theory, preprint.*

Appendix A

The program is listed below and should be saved as a *.sage* file. There is also a second program that runs in Sage using Pyrex. It should be saved as a *.spyx* file. Pyrex is a compiler language that is used to speed up some of the calculations in the loops of the main "race1" function. If the these calculations were done in Sage, the program would run about eight times slower on average. To run it, load both files in sage and then enter race1(k,T,ustart,uend,step) into the console with the k, T, *ustart*, *uend*, and *step* replaced with the desired values.

```
"""
Use zeros of Dirichlet L-functions to prove that
pi(x,q,a)-pi(x,q,b) has infinitely many sign changes

by Jason Sneed (2007)

"""
##########

def quad(k):
    """
    Return list of quadratic residues and non-residues

    INPUT:
        k -- positive integer

    EXAMPLES:
        sage: quad(24)
        ([1], [5, 7, 11, 13, 17, 19, 23])
        sage: quad(11)
        ([1, 3, 4, 5, 9], [2, 6, 7, 8, 10])

    """
    res=[]
    for i in range(k):
```

```
        if gcd(i,k) == 1:
    res.append(i)
    qr=quadratic_residues(k)
    qr1=[]
    for i in qr:
        if gcd(i,k)==1:
            if i>1:
                qr1.append(i)
    nqr=[]
    for i in res:
        count=0
        for j in qr:
            if i==j:
                count=count+1
        if count == 0:
            nqr.append(i)
    return qr1,nqr

##########

def Dir_Lfunc_zeros(k,T):
    """
    Obtain list of zeros for Dirichlet L-functions to
    modulus k, up to height T.  Groups conjugate L-functions together.

    INPUT:
        k -- positive integer
        T -- positive real number

    EXAMPLES:
sage: get_zeros(5,10)
        ([[-4.1329037052100004,
           6.18357819545,
           8.4572291744200001,
           -9.4429311297300007],
          [6.64845334473, 9.8314444328900006]],
         2)

    """
    H=floor(T*log(k*T/(2*pi*e))/pi+.9185*log(k*T)+5.512)
# upper bound for number of zeros to height T, by K. McCurley (1984)

    S=''.join(['-z ',`H`,' -a -s ',`k`,' -f ',`k`])
    L = lcalc(S)
    L = L.split()
    oval = []     ## Creates multi-dimensional list of the zeros < T
    count = 0     ## called oval to using Rubinstein'l lcalc program
    ind=[]
    lengL=int(len(L)/3-1)
    for i in range(lengL):
        flag=0
```

```
    t=int(L[3*i+1])
    if t > 1:
        if len(ind)==0:
            ind.append(t)
            oval.append([])
            oval[count].append(float(L[3*i+2]))
            flag=1
        if flag == 0:
            if t == int(ind[count]):
                if abs(float(L[3*i+2])) <= T:
                    oval[count].append(float(L[3*i+2]))
            else:
                count=count+1
                oval.append([])
                oval[count].append(float(L[3*i+2]))
                ind.append(t)

F=divisors(k)
ovalRu={}    ## ovalRu is a list of zeros created by reading
j=0          ## in files of Rumely zeros names ZERO35.txt (here
counter = 0  ## k=35).  The Rumely include only the zeros from
for j in F:  ## primitive characters mod k so multiple files
    if j > 2:    ## be read in usually
        if is_odd(j):
            counter=len(ovalRu)
            zerofilename = ''.join(['ZERO','j', '.txt'])
            zfl=open(zerofilename,"r")
            mess=zfl.read()
            mess=mess.split()
            leng=int(len(mess)/3-1)
            for i in range(leng):
                t=int(mess[3*i+1])-1+counter
                if ovalRu.has_key(t):
                    if abs(float(mess[3*i+2])) <= T:
                        ovalRu[t].append(float(mess[3*i+2]))
                else:
                    ovalRu[t]=[float(mess[3*i+2])]
        if j/4 - floor(j/4) < .000001:
            counter=len(ovalRu)
            zerofilename = ''.join(['ZERO','j', '.txt'])
            zfl=open(zerofilename,"r")
            mess=zfl.read()
            mess=mess.split()
            leng=int(len(mess)/3-1)
            for i in range(leng):
                t=int(mess[3*i+1])-1+counter
                if ovalRu.has_key(t):
                    if abs(float(mess[3*i+2])) <= T:
                        ovalRu[t].append(float(mess[3*i+2]))
                else:
                    ovalRu[t]=[float(mess[3*i+2])]
```

```
    oval1=[]              ## oval1 will be a list that contains the
    lengo=len(oval)       ## the zeros in ovalRu but organized in the
    lengR=len(ovalRu)     ## same way as the Rubinstein zeros in oval.
    for i in range(lengo):   ## A matching scheme between oval and
        flag = 0          ## ovalRu is used to properly rearrange
        flag1 = 0         ## zeros into oval1.
        count = 0
        count1 = 0
        oval1.append([])
        for j in oval[i]:
            if flag == 0:
                if j > 0:
                    for r in range(lengR):
                        for w in ovalRu[r]:
                            if abs(j-w)<.00001:
                                for y in ovalRu[r]:
                                    oval1[i].append(y)
                                flag = 1
                                count = count + 1
                            if count > 1:
                                print "Too many matches"
                                print i, j
                    if count == 0:
                        print "No Match"
                        print i, j
            if flag1 == 0:
                if j < 0:
                    for r in range(lengR):
                        for w in ovalRu[r]:
                            if abs(j+w)<.00001:
                                for y in ovalRu[r]:
                                    oval1[i].append(-y)
                                flag1 = 1
                                count1 = count1 + 1
                            if count1 > 1:
                                print "Too many matches"
                                print i
                    if count1 == 0:
                        print "No Match"
                        print i,j

    return oval1

#############

def char_values_zeros(k,T,zlist):
    """
    Obtain list of Dirichlet character values for modulus k,
    and list of zeros for each character.
    Separate list for real characters and complex characters.
```

```
INPUT:
    k -- positive integer
    zlist -- list of zeros, output from char_zeros() above
EXAMPLES:
    sage:
"""
import math
pi=math.pi
cos=math.cos
sin=math.sin
dir=[]
import re
R=''.join(['-a -s ',`k`,' -f ',`k`,' -C 1'])
L = lcalc(R)        ## Uses lcalc to calculate char values
L = L.split('\n')
r = re.compile('[0-9]+ [0-9]+ [0-9]+ [0-9]+ +')
count=0
temp=2
l = 0
while l < len(L):
    if r.match(L[l]) <> None:
        t = L[l].split(' ')
        ch = int(t[1])
        if ch > temp:
            count = count+1
            temp=ch
        if ch <> 1:
            if count>len(dir)-1:
                dir.append([])
                for j in range(k):
                    dir[count].append([0,0])
            mod=int(t[3])
            dir[count][mod][0]=float(t[4])
            dir[count][mod][1]=float(t[5])
    l=l+1


rechar=[]
imchar=[]
reval=[]
imval=[]
counti=0     # number of complex characters
countr=0     # number of real characters
recalc=[]
imcalc=[]
for g in range(len(dir)):
    count=0
    for j in range(k):
        if abs(dir[g][j][1]) < .000000001:
            count=count+1
```

```
        if count==k:
            rechar.append([])
            recalc.append([])
            for j in range(k):
                t=dir[g][j][0]
                rechar[countr].append(t)
            reval.append(zlist[g])
            for y in zlist[g]:
                q=((1-abs(y)/T)*cos((pi)*y/T)+sin((pi)*abs(y)/T)/pi)*(1/4+y**2)**(-1)
                recalc[countr].append(q)  ## performing calculations
            countr=countr+1         ## to be used in main func later.
        else:
            imchar.append([])
            imcalc.append([])
            for j in range(k):
                t=dir[g][j]
                imchar[counti].append(t)
                s=zlist[g]
            imval.append(s)
            for y in zlist[g]:
                q=((1-abs(y)/T)*cos((pi)*y/T)+sin((pi)*abs(y)/T)/pi)*(1/4+y**2)**(-1)
                imcalc[counti].append(q)  ## performing calculations
            counti=counti+1             ## to be read into main func.
    return rechar,reval,countr,imchar,imval,counti,recalc,imcalc

######################################################
#
#  Check all races modulo k
#  quadratic residue (including 1) vs. quadratic non-residue
#
######################################################

def race1(k,T,ustart,uend,step):
    """
    Run the prime races modulo k using zeros of L-functions to height T.
    Evaluate AT* function at points u between ustart and uend, step is
       the increment.

    INPUT:
        k -- positive integer.
        T -- positive real number.
        ustart, uend, step - real numbers
    """
    import math
    pi=math.pi
    cos=math.cos
    sin=math.sin
    oval1=Dir_Lfunc_zeros(k,T)  ## reading in zeros and char values
    rechar,reval,countr,imchar,imval,counti,recalc,imcalc=char_values_zeros(k,T,oval1)
    qr1,nqr=quad(k)
    count=0
```

```
    w=len(qr1)+1
    v=len(nqr)
    N=(w+v)/w
    print "modulus=",k,"   T=",T,"    N=",N
    y=(w-1)*v
    print "y=",y
    for p in qr1:
        l1=p
        for q in nqr:
           l2=q
           print "Racing ",l1," vs. ",l2
           s=[]
           good=0
           bestsum=-1000.0;
           for x in range(floor((uend-ustart)/step)+1):
               u=ustart+x*step
               sum=0  ## sum = A*T(u) function.
               sum=imsum(countr,rechar,reval,recalc,counti,imchar,l1,l2,imval,u,imcalc,sum)
               s.append([u,sum])
               if (sum>bestsum):
                   bestsum=sum  ## prints new high sum until success
                   print "u=",u,"   sum=",sum
               if (sum > N + .05) and (good==0):
                   print "Success"         ## Alerts of successful race
                   print "***************"
                   count=count+1
                   good=1
                   break
           if good==0:
               print "Test failed. Try changing T, ustart, uend and/or steps."
               break
           if count == y:
               print "All races for this modulus have infinitely many sign changes!"


###########
```

The Pyrex Code

```
def imsum(countr,rechar,reval,recalc,counti,imchar,l1,l2,imval,u,imcalc,sum):
    import math
    sin = math.sin
    cos = math.cos
    for n in range(counti):
        a = imchar[n][l1][0]-imchar[n][l2][0]
        b = imchar[n][l1][1]-imchar[n][l2][1]
        for i in range(len(imval[n])):
            sum = sum-imcalc[n][i]*((a-2*b*imval[n][i])*cos(imval[n][i]*u)+
\(2*a*imval[n][i]+b)*sin(imval[n][i]*u))
    for n in range(countr):
        a = rechar[n][l1]-rechar[n][l2]
        for i in range(len(reval[n])):
```

```
            sum = sum-recalc[n][i]*(a*cos(reval[n][i]*u)+(2*reval[n][i]*a)*
\sin(reval[n][i]*u))
    return sum
```

Jason Sneed
Department of Mathematics
University of Illinois at Urbana-Champaign
Urbana, IL 61801, U.S.A.
Email: jpsneed@math.uiuc.edu