Experimental Mathematics, 21(3):307–328, 2012

ISSN: 1058-6458 print / 1944-950X online
DOI: 10.1080/10586458.2012.687238

Taylor & Francis
Taylor & Francis Group

Conjectures and Experiments Concerning the Moments of $L(1/2, \chi_d)$

Matthew W. Alderson and Michael O. Rubinstein

CONTENTS

We report on some extensive computations and experiments concerning the moments of quadratic Dirichlet L-functions at the critical point. We computed the values of $L(1/2, \chi_d)$ for $-5 \times 10^{10} < d < 1.3 \times 10^{10}$ in order to numerically test conjectures concerning the moments $\sum_{|d|<X} L(1/2, \chi_d)^k$. Specifically, we tested the full asymptotics for the moments conjectured by Conrey, Farmer, Keating, Rubinstein, and Snaith, as well as the conjectures of Diaconu, Goldfeld, Hoffstein, and Zhang concerning additional lower-order terms in the moments. We also describe the algorithms used for this large-scale computation.

1. INTRODUCTION

Let D be a square-free integer, $D \neq 0, 1$, and let $K = \mathbb{Q}(\sqrt{D})$ be the corresponding quadratic field. The fundamental discriminant d of K equals D if $D = 1 \mod 4$, and $4D$ if $D = 2, 3 \mod 4$. Let $\chi_d(n)$ be the Kronecker symbol $\left(\frac{d}{n}\right)$, and $L(s, \chi_d)$ the quadratic Dirichlet L-function given by the Dirichlet series

$$L(s, \chi_d) = \sum_{n=1}^{\infty} \frac{\chi_d(n)}{n^s}, \quad \Re(s) > 0,$$

satisfying the functional equation

$$L(s, \chi_d) = |d|^{\frac{1}{2}-s} X(s, a) L(1-s, \chi_d),$$

where

$$X(s, a) = \pi^{s-\frac{1}{2}} \frac{\Gamma\left(\frac{1-s+a}{2}\right)}{\Gamma\left(\frac{s+a}{2}\right)}, \quad a = \begin{cases} 0 & \text{if } d > 0, \\ 1 & \text{if } d < 0. \end{cases} \tag{1–1}$$

In this paper, we describe some experiments concerning the moments of quadratic Dirichlet L-functions at the central critical point:

$$\sum_{d \in D(X)} L(1/2, \chi_d)^k. \tag{1–2}$$

Here, k is a positive integer, and $D(X)$ denotes the set of fundamental discriminants with $|d| < X$.

2000 AMS Subject Classification: 11M06
Keywords: moments of $L(1/2, \chi_d)$

Several conjectures exist for these moments. For instance, in [Keating and Snaith 00], motivated by the fundamental work of [Katz and Sarnak 99] and based on an analogous result in random matrix theory, the authors of that paper conjectured a formula for the leading asymptotics of (1–2). Specifically, they conjectured that as $X \to \infty$,

$$\frac{1}{|D(X)|}\sum_{d\in D(X)} L\left(\frac{1}{2},\chi_d\right)^k \sim a_k \prod_{j=1}^{k}\frac{j!}{(2j)!}\log(X)^{k(k+1)/2}, \tag{1–3}$$

where a_k is an arithmetic factor, described in [Conrey and Farmer 00], of the form

$$a_k = \prod_p \frac{\left(1-\frac{1}{p}\right)^{k(k+1)/2}}{1+\frac{1}{p}} \times \left(\frac{\left(1-\frac{1}{\sqrt{p}}\right)^{-k}+\left(1+\frac{1}{\sqrt{p}}\right)^{-k}}{2}+\frac{1}{p}\right).$$

In a few cases, Keating and Snaith's conjecture agrees with known theorems, e.g., for $k = 1, 2$ [Jutila 81], and for $k = 3$ [Soundararajan 00].

Subsequently, a more precise asymptotic expansion for (1–2) was predicted by Conrey, Farmer, Keating, Rubinstein, and Snaith (CFKRS) [Conrey et al. 05]. The lower-order terms are affected by the form of the gamma factors in the functional equation for $L(s, \chi_d)$, so naturally the authors considered the subset of $d > 0$ separately from $d < 0$. Therefore, let

$$D_+(X) = \{d \in D(X) : d > 0\},$$
$$D_-(X) = \{d \in D(X) : d < 0\}.$$

The conjecture of CFKRS states that

$$\sum_{d\in D\pm(X)} L\left(\frac{1}{2},\chi_d\right)^k \sim \frac{3}{\pi^2} X \mathcal{Q}_\pm(k,\log X), \tag{1–4}$$

where $\mathcal{Q}_\pm(k, x)$ is a polynomial of degree $k(k + 1)/2$ in x. The fraction $3/\pi^2$ accounts for the density of fundamental discriminants among all the integers. In their paper, CFKRS also conjectured a remainder term of size $O(X^{1/2+\epsilon})$, but the evidence, both theoretical and numerical (discussed below), suggests the existence of lower-order terms for $k \geq 3$.

The beauty in their conjecture lies in the fact that it gives a formula for the polynomial in question. The polynomial $\mathcal{Q}_\pm(k, \log X)$ is expressed in terms of a more fundamental polynomial $Q_\pm(k, x)$ of the same degree that captures the moments locally:

$$\mathcal{Q}_\pm(k,\log X) = \frac{1}{X}\int_1^X Q_\pm(k,\log t)dt.$$

The polynomial $Q_\pm(k, x)$ is described below, and its leading coefficient agrees with the conjecture of Keating and Snaith (1–3).

The polynomial $Q_\pm(k, x)$ of CFKRS is given by them as the k-fold residue

$$Q_\pm(k,x) = \frac{(-1)^{k(k-1)/2}2^k}{k!}\frac{1}{(2\pi i)^k}\oint\cdots \tag{1–5}$$
$$\oint \frac{G_\pm(z_1,\ldots,z_k)\Delta(z_1^2,\ldots,z_k^2)^2}{\prod_{j=1}^k z_j^{2k-1}} e^{\frac{x}{2}\sum_{j=1}^k z_j}\,dz_1\ldots dz_k,$$

where

$$G_\pm(z_1,\ldots,z_k) = A_k(z_1,\ldots,z_k) \times \prod_{j=1}^k X\left(\frac{1}{2}+z_j,a\right)^{-1/2}\prod_{1\leq i\leq j\leq k}\zeta(1+z_i+z_j)$$

and

$$\Delta(z_1^2,\ldots,z_k^2) = \prod_{1\leq i<j\leq k}(z_j^2-z_i^2)$$

is a Vandermonde determinant. Here, $a = 0$ for G_+ and $a = 1$ for G_-, $X(s, a)$ is given in (1–1), and A_k equals the Euler product, absolutely convergent for $|\Re z_j| < 1/2$, defined by

$$A_k(z_1,\ldots,z_k) = \prod_p \prod_{1\leq i\leq j\leq k}\left(1-\frac{1}{p^{1+z_i+z_j}}\right) \times \left(\frac{1}{2}\left(\prod_{j=1}^k\left(1-\frac{1}{p^{1/2+z_j}}\right)^{-1} + \prod_{j=1}^k\left(1+\frac{1}{p^{1/2+z_j}}\right)^{-1}\right)+\frac{1}{p}\right)\times\left(1+\frac{1}{p}\right)^{-1}.$$

More generally, CFKRS predicted that for suitable weight functions g,

$$\sum_{d\in D_\pm(\infty)} L(1/2,\chi_d)^k g(|d|) \sim \frac{3}{\pi^2}\int_1^\infty Q_\pm(k,\log t)g(t)dt. \tag{1–6}$$

The method that CFKRS used to heuristically derive this formula relies on number-theoretic techniques, specifically the approximate functional equation, but they were guided by analogous results in random matrix theory to help determine the form of the conjecture.

An alternative approach to conjecturing moments exists. In their paper [Diaconu et al. 03], Diaconu, Goldfeld, and Hoffstein (DHG) used the double Dirichlet

series

$$Z_k(s,w) = \sum_{d \in D(\infty)} \frac{L(s,\chi_d)^k}{|d|^w}$$

to study the moments of $L(1/2, \chi_d)$. In particular, they showed how one can derive a formula for the cubic moments of $L(1/2, \chi_d)$ by investigating the polar behavior of $Z_3(s,w)$.

The method of DGH produces a proof for the cubic moment of $L(1/2, \chi_d)$. Specifically, they show that the difference of both sides of (1–4) for $k = 3$ is, for every $\epsilon > 0$, of size $O_\epsilon(X^{\theta+\epsilon})$, where $\theta = 0.85366\ldots$. They also give a remainder term for a smoothed cubic moment with $\theta = 4/5$. Recently, Young has obtained an improved estimate for the remainder term of size $O(X^{3/4+\epsilon})$ [Young 12]. The moments he considered were smoothed, and for simplicity, he considered the subset of discriminants divisible by 8 and positive. The appearance of $3/4 + \epsilon$ is very interesting in light of the next paragraph.

The method of DGH predicts the existence of a further lower-order term of size $X^{3/4}$. In particular, DGH conjectured that there exists a constant b such that

$$\sum_{d \in D(X)} L\left(\frac{1}{2}, \chi_d\right)^3 = \frac{6}{\pi^2} X \mathcal{Q}(3, \log X) + bX^{3/4} + O\left(X^{1/2+\epsilon}\right). \tag{1–7}$$

The existence of such a term comes from a pole of the double Dirichlet series at $w = 3/4$ and $s = 1/2$, the conjectured meromorphic continuation of $Z_3(1/2, w)$ to $\Re(w) < 3/4$, and assumes a growth condition on $Z_3(1/2, w)$.

DGH also suggest that additional lower-order terms, infinitely many for each $k \geq 4$, are expected to persist. The form of these terms is described in the survey [Zhang 06], along with an exposition of the approach using double Dirichlet series. Their conjecture involving lower-order terms is stated in the following form. For $k \geq 4$ and every $\epsilon > 0$,

$$\sum_{d \in D(X)} L\left(\frac{1}{2}, \chi_d\right)^k = \sum_{l=1}^{\infty} X^{(l+1)/(2l)} P_l(\log x) + O(X^{1/2+\epsilon}), \tag{1–8}$$

where every P_l is a polynomial depending on k. In particular, P_1 is a polynomial of degree $k(k+1)/2$, presumably agreeing with the polynomial predicted by the CFKRS conjecture, but to our knowledge this agreement has not been checked.

Zhang further conjectured [Zhang 05] that $b \approx -0.2154$, and in a private communication to one of the authors, reported that he also computed the constants associated with the $X^{3/4}$ term when one restricts to $d < 0$ or to $d > 0$, thus predicting

$$\sum_{d \in D_\pm(X)} L\left(\frac{1}{2}, \chi_d\right)^3 = \frac{3}{\pi^2} X \mathcal{Q}_\pm(3, \log X) + b_\pm X^{3/4} + O\left(X^{1/2+\epsilon}\right),$$

with $b_+ \approx -0.14$ and $b_- \approx -0.07$ (note that $b = b_+ + b_-$). His evaluation of b, b_+, and b_- involves an elaborate sieving process, and also depends on unproven hypotheses regarding the meromorphic continuation and rate of growth of Z_3.

While it might seem that a term as large as $X^{3/4}$ in the cubic moment should be easily detected, two things make it very challenging in this context: the small size of the constants involved, and also the fact that the remainder term, conjecturally of size $O(X^{1/2+\epsilon})$, dominates even in our large data set; presumably the X^ϵ can get as large as some power of $\log(x)$, which can dominate $X^{1/4}$ even for values of X as large as 10^{11}.

For this reason, we embarked on a large-scale computation in order to see whether such a lower-order main term in the cubic moment could be detected. We also carried out extensive verification of the predictions of CFKRS for $k = 1, \ldots, 8$. While CFKRS provided some modest data in [Conrey et al. 05] for $|d| < 10^7$, we carried out tests for $-5 \times 10^{10} < d < 0$ and $0 < d < 1.3 \times 10^{10}$. In order to dampen the effect of the noisy remainder term, we also considered smoothed moments.

Our numerical results are described in Section 2. Interestingly, they lend support to both the full asymptotic expansion conjectured by CFKRS and to the existence of lower-order terms predicted by DGH and Zhang.

In Section 3, we describe the two methods that we used to compute a large number of values of $L(1/2, \chi_d)$ for $d < 0$ and separately for $d > 0$. The first, for $d < 0$, is based on the theory of binary quadratic forms, and uses Chowla and Selberg's K-Bessel expansion of the Epstein zeta function [Chowla and Selberg 67]. The second, for $d > 0$, uses a traditional smooth approximate functional equation. Both methods have comparable runtime complexities, but the former has the advantage of being faster by a constant factor. See the end of the paper for a discussion that compares the two runtimes.

2. NUMERICAL DATA

In this section, we numerically examine the conjectures of CFKRS, DGH, and Zhang for the moments of $L(1/2, \chi_d)$.

The collected data provide further evidence in favor of the CFKRS conjecture concerning the full asymptotics of the moments of $L(1/2, \chi_d)$. With respect to the remainder term, the numerics also seem to suggest the presence of additional lower-order terms as predicted by DGH and Zhang.

In Tables 1 and 2 and Figures 1 to 4 we depict the quantities

$$R_{\pm}(k,X) := \sum_{d \in D_{\pm}(X)} L\left(\frac{1}{2}, \chi_d\right)^k \Big/ \frac{3}{\pi^2} \int_1^X Q_{\pm}(k, \log t) dt, \tag{2–1}$$

and the related difference

$$\Delta_{\pm}(k,X) := \sum_{d \in D_{\pm}(X)} L\left(\frac{1}{2}, \chi_d\right)^k - \frac{3}{\pi^2} \int_1^X Q_{\pm}(k, \log t) dt, \tag{2–2}$$

for $k = 1, \ldots, 8$ and both positive and negative discriminants d.

The quantity (2–1) measures the consistency of CFKRS's prediction, while (2–2) allows one to see the associated remainder term. The numerator of (2–1) was calculated by computing many values of $L(1/2, \chi_d)$, using the methods described in the next two sections. The denominator was obtained from numerically approximated values of the coefficients of $Q_{\pm}(k, \log t)$, computed in the same manner as that performed in [Conrey et al. 05], though to higher precision. Tables of the coefficients of the polynomials $Q_{\pm}(k, x)$ can be found in [Conrey et al. 05]. These values were then also used in graphing the difference (2–2).

Tables 1 and 2 provide strong numerical support in favor of the asymptotic formula predicted by CFKRS, described in equations (1–4) and (1–5), for both $d < 0$ and $d > 0$, agreeing to seven to eight decimal places for $k = 1$, and four to five decimal places for $k = 8$.

In the figures we depict thousands of values of the quantities (2–1) and (2–2) at multiples of 10^7, i.e., $X = 10^7, 2 \times 10^7, \ldots$. We display data up to $X = 1.3 \times 10^{10}$ for $d > 0$, and $X = 5 \times 10^{10}$ for $d < 0$. The larger amount of data for $d < 0$ reflects the faster method that we used for computing the corresponding L-values.

In Figures 1 and 2, notice that each graph fluctuates tightly about 1, with the extent of fluctuation becoming progressively larger as k increases, as indicated by the varying vertical scales. The graphs show excellent agreement with the full asymptotics as predicted by CFKRS across all eight moments computed, for both $d < 0$ and $d > 0$. One also notices a slight downward shift from 1 in the $k = 3$ plots, as predicted by DGH and Zhang.

We also depict in Figures 3 and 4 the differences (2–2) as dots, as well as the running average of the plotted differences as a solid curve. While we plot the average every 10^7, these running averages were computed by sampling the differences every 10^6. We chose to display one-tenth of the computed values in order to make our plots more readable given the limited resolution of computer displays and printers.

Averaging has the effect of smoothing the moment and reducing the impact of the noisy remainder term. It allows one to see more clearly whether there are any biases hiding within the noise. This running average gives a discrete approximation to the smoothed difference:

$$\Delta_{\pm}(k,X) := \sum_{d \in D_{\pm}(X)} L\left(\frac{1}{2}, \chi_d\right)^k \left(1 - \frac{|d|}{X}\right) - \frac{3}{\pi^2} \int_1^X Q_{\pm}(k, \log t) \left(1 - \frac{t}{X}\right) dt. \tag{2–3}$$

We make several observations concerning Figures 3 and 4. First, for $k = 1$, the observed remainder term is seemingly of size $X^{1/4+\epsilon}$, much smaller than Goldfeld and Hoffstein's proven bound of $O(X^{19/32+\epsilon})$ for the first moment, and also smaller than the bound of $O(X^{1/2+\epsilon})$ implied in their work for a smoothed first moment (for the latter, see also [Young 09]). Furthermore, there appears to be a bias in the average remainder term. This is especially apparent for $d < 0$ and further supported by our log log plots in Figure 6.

For $k = 2$, the remainder term, even when averaged, fluctuates above and below 0. The largest average remainder for $k = 2$ in our data set was of size roughly 1.3×10^4, consistent with a conjectured remainder, for $k = 2$, of size $O(X^{1/2+\epsilon})$.

In Figures 3 and 4, a bias of the kind predicted by DGH can be seen. For $k \geq 3$, a noticeable bias is evident in the remainder term, especially when averaged, and most prominently for $d > 0$.

We elaborate on this last point further. In the case of $k = 3$, DGH predict a single main lower-order term of the form $bX^{3/4}$, and as described in the introduction, Zhang worked out the value of b, separately for $d < 0$ and $d > 0$, as equal to -0.07 and -0.14 respectively.

One possible explanation for the fact that the bias appears more prominently for $d > 0$, even though we

k	$\sum_{d\in D_-(X)} L(1/2,\chi_d)^k$	$\frac{3}{\pi^2}\int_1^X Q_-(k,\log t)dt$	$R_-(k,x)$	$\Delta_-(k,X)$
1	25458527125.376	25458526443.085	1.0000000268001	682.291
1	52401254983.398	52401252573.351	1.0000000459922	2410.047
1	79904180421.746	79904180600.902	0.99999999775786	−179.156
1	107770905413.09	107770904521.07	1.0000000082770	892.02
1	135908144579.9	135908144595.65	0.99999999988411	−15.75
2	695798091128.96	695797942880.62	1.0000002130623	148248.34
2	1505736931971.7	1505736615082.0	1.0000002104549	316889.7
2	2362905062077.2	2362905209666.9	0.99999993753888	−147589.7
2	3251727763805.6	3251727486319.2	1.0000000853351	277486.4
2	4164586513531.5	4164586544704.8	0.99999999251467	−31173.3
3	35923488939396.	35923434720074.	1.0000015093023	54219322.
3	82792501873632.	82792433101707.	1.0000008306547	68771925.
3	0.13470723693602e15	0.13470723096090e15	1.0000000443563	5975116
3	0.19013982678941e15	0.19013979175101e15	1.0000001842770	35038394
3	0.24831500039182e15	0.24831501538879e15	0.99999993960505	−14996973
4	0.26221677201508e16	0.26221542614856e16	1.0000051326749	13458665240
4	0.64846065425230e16	0.64845918799277e16	1.0000022611439	14662595290
4	0.10987196470794e17	0.10987187884822e17	1.0000007814531	0.8585972e10
4	0.15956123181403e17	0.15956125546013e17	0.99999985180550	−0.2364610e10
4	0.21299535514803e17	0.21299540911015e17	0.99999974665125	−0.5396212e10
5	0.23541937472178e18	0.23541622006477e18	1.0000134003384	0.315465701e13
5	0.62771726711464e18	0.62771414322685e18	1.0000049766089	0.312388779e13
5	0.11106890853615e19	0.11106862772711e19	1.0000025282480	0.28080904e13
5	0.16628632428499e19	0.16628683849741e19	0.99999690767817	−0.51421242e13
5	0.22724025077610e19	0.22724048423231e19	0.99999897264693	−0.23345621e13
6	0.24225487162243e20	0.24224780818937e20	1.0000291578822	0.706343306e15
6	0.69880224640908e20	0.69879554487455e20	1.0000095901220	0.670153453e15
6	0.12937968210632e21	0.12937887586288e21	1.0000062316467	0.80624344e15
6	0.19996752978479e21	0.19997013306315e21	0.99998698166411	−0.260327836e16
6	0.28005925088677e21	0.28006019455853e21	0.99999663046810	−0.94367176e15
7	0.274712571777e22	0.274697762672e22	1.00005391054	0.14809105e18
7	0.859431066562e22	0.859415893116e22	1.00001765553	0.15173446e18
7	0.166743403869e23	0.166740957095e23	1.00001467410	0.2446774e18
7	0.266330275024e23	0.266339641978e23	0.999964830793	−0.9366954e18
7	0.382588166641e23	0.382591322018e23	0.999991752617	−0.3155377e18
8	0.3351697755e24	0.3351406841e24	1.000086804	0.290914e20
8	0.1139465805e25	0.1139429048e25	1.000032259	0.36757e20
8	0.2319359069e25	0.2319282301e25	1.000033100	0.76768e20
8	0.3831454627e25	0.3831738559e25	0.9999259000	−0.283932e21
8	0.5649093016e25	0.5649183210e25	0.9999840342	−0.90194e20

TABLE 1. Moments $\sum_{d\in D_-(X)} L(1/2,\chi_d)^k$ versus CFKRS's $\frac{3}{\pi^2}\int_1^X Q_-(k,\log t)dt$, for $k = 1,\ldots,8$ and $d < 0$. Five values for each k are shown, at $X = 10^{10}, 2\times 10^{10},\ldots,5\times 10^{10}$.

have less data in that case, is that the "noise" appears numerically to be much larger for $d < 0$. Comparing Figure 3 with Figure 4, we notice that the plotted remainder terms seem to be about ten times larger for $d < 0$ as compared to $d > 0$, even when restricted to $|d| < 1.3\times 10^{10}$. A larger amount of "noise" in the remainder term makes it harder to detect a lower-order term hiding within the noise, especially when the lower-order terms are married to such small constant factors: $-0.07X^{3/4}$ and $-0.14X^{3/4}$, as predicted by Zhang, before averaging, and $4/7$ as large after averaging over X.

k	$\sum_{d\in D_+(X)} L(1/2,\chi_d)^k$	$\frac{3}{\pi^2}\int_1^X Q_+(k,\log t)dt$	$R_+(k,x)$	$\Delta_+(k,X)$
1	4074391863.4447	4074392042.9388	0.99999995594580	−179.4941
1	8445624718.0243	8445624023.3138	1.0000000822569	694.7105
1	12928896894.590	12928896383.146	1.0000000395582	511.444
1	17484928279.579	17484927921.500	1.0000000204793	358.079
1	22095062063.114	22095062690.738	0.99999997159438	−627.624
2	76310075816.466	76310057832.320	1.0000002356720	17984.146
2	168051689378.93	168051603484.03	1.0000005111222	85894.90
2	266303938917.29	266303916920.62	1.0000000825999	21996.67
2	368948427173.22	368948308826.37	1.0000003207681	118346.85
2	474942139636.16	474942177549.68	0.99999992017235	−37913.52
3	2478393690176.2	2478391641054.5	1.0000008267950	2049121.7
3	5878735240405.9	5878729153410.4	1.0000010354271	6086995.5
3	9720154390088.4	9720158187579.5	0.99999960931797	−3797491.1
3	13873264940982.	13873252832529.	1.0000008727912	12108453.
3	18271480140004.	18271496263135.	0.99999911758015	−16123131.
4	0.10868425484737e15	0.10868409751016e15	1.0000014476562	157337203
4	0.27974980520169e15	0.27974915668497e15	1.0000023182079	648516719
4	0.48473276073219e15	0.48473329605692e15	0.99999889563038	−535324726
4	0.71493167429315e15	0.71492961664246e15	1.0000028781164	2057650687
4	0.96564046289913e15	0.96564334647659e15	0.99999701382764	−2883577466
5	0.57022430562904e16	0.57022322406897e16	1.0000018967310	10815600670
5	0.15999737676260e17	0.15999653478756e17	1.0000052624580	0.84197504e11
5	0.29130430291967e17	0.29130495012249e17	0.99999777826357	−0.64720282e11
5	0.44482716417300e17	0.44482376920928e17	1.0000076321545	0.339496372e12
5	0.61707290890367e17	0.61707708869778e17	0.99999322646362	−0.417979411e12
6	0.33658290814098e18	0.33658163201404e18	1.0000037914337	0.127612694e13
6	0.10326933113376e19	0.10326816848898e19	1.0000112585010	0.116264478e14
6	0.19792425806612e19	0.19792515491256e19	0.99999546875969	−0.89684644e13
6	0.31332379844474e19	0.31331890401641e19	1.0000156212353	0.489442833e14
6	0.44685941512069e19	0.44686487402482e19	0.99998778399367	−0.545890413e14
7	0.215991539086e20	0.215989246213e20	1.00001061568	0.2292873e15
7	0.726312167992e20	0.726295668031e20	1.00002271797	0.16499961e16
7	0.146733199900e21	0.146734533114e21	0.999990914109	−0.1333214e16
7	0.241042340834e21	0.241036160843e21	1.00002563927	0.6179991e16
7	0.353694078736e21	0.353700808054e21	0.999980974547	−0.6729318e16
8	0.1475899774e22	0.1475859642e22	1.000027192	0.40132e17
8	0.5449090667e22	0.5448853612e22	1.000043505	0.237055e18
8	0.1161602962e23	0.1161622793e23	0.9999829282	−0.19831e18
8	0.1981618159e23	0.1981550523e23	1.000034133	0.67636e18
8	0.2993403001e23	0.2993484649e23	0.9999727248	−0.81648e18

TABLE 2. Moments $\sum_{d\in D_+(X)} L(1/2,\chi_d)^k$ versus $\frac{3}{\pi^2}\int_1^X Q_+(k,\log t)dt$, for $k=1,\dots,8$ and $d>0$. Five values for each k are shown, $X = 2\times 10^9, 4\times 10^9, \dots, 10^{10}$.

Thus, even though one expects, in the long run, to see an $X^{3/4}$ term dominate over noise of size $X^{1/2+\epsilon}$, in the ranges examined it seems that the noise has a large impact.

Another factor possibly negatively affecting the quality of the lower-order term detected when $d<0$ is that the predicted constant factor is about one-half as large: -0.07 for $d<0$, compared to -0.14 for $d>0$. Combined with noise that is ten times bigger, it is not surprising that the quality of the average remainder term for $d<0$ seems to be more affected by the noise.

In Figure 5 we redisplay the $k = 3$ plots from Figures 3 and 4, zoomed in to allow one to see the average remainder term in greater detail. Here we also depict the prediction of Zhang (dashed line). More precisely, the dashed line represents the average predicted lower-order term:

$$\frac{1}{X}\int_0^X b_{\pm} t^{3/4} dt = \frac{4}{7} b_{\pm} x^{3/4}, \tag{2–4}$$

with $b_- = -0.07$ for $d < 0$ and $b_+ = -0.14$ for $d > 0$. For $d > 0$, the fit of the average remainder term against Zhang's prediction is very nice. For $d < 0$, the plot supports a bias in the sense that the average value is mainly negative, but the fit against $-0.07\frac{4}{7}X^{3/4}$ is far from conclusive.

Plots of the average remainder term on a log log scale are shown in Figure 6 for $1 \le k \le 4$ and positive/negative d. On a log log scale, a function of X of the form $f(X) = BX^{3/4}$ is transformed into the function of $u = \log X$ given by $\log B + 3/4u$, i.e., a straight line with slope $3/4$. We compare, in the $k = 3$ plots, the log log plot of the average remainder against Zhang's prediction. The fit is especially nice for $d > 0$, but only in crude qualitative terms for $d < 0$, consistent with the observations made concerning the poorer fit in the $k = 3$, $d < 0$, plots of Figures 3–5.

For $k \ge 4$, DGH predict infinitely many lower-order terms, with the largest one of size $X^{3/4}$ times a power of $\log X$ that they did not make explicit. Interestingly, a bias in support of this does appear evident, especially for $k = 4$ and $d > 0$. In that log log plot, the remainder term does seem reasonably straight, suggesting a lower-order term obeying a power law, perhaps with some additional powers of log.

It is reasonable to contest that the observed biases here exist due to a persistent small error in the calculation of the moment polynomials or in the values of $L(1/2, \chi_d)$. In an effort to alleviate such concerns, the computations yielding our numerics were executed again, in a limited way, using higher precision. As anticipated, these higher-precision results remained consistent with the initial results, reducing the possibility of such a bias existing. Furthermore, the overall excellent agreement of the computed moments with the predicted asymptotic formula of CFKRS supports the correctness of the computation.

3. OUR COMPUTATIONAL FORMULAS

The computations for the moments of $L(1/2, \chi_d)$ hinge on the efficient computation of $L(1/2, \chi_d)$ itself for many discriminants d. This computation is split into two cases according to whether d is positive or negative. In the former case, we calculate $L(1/2, \chi_d)$ using a smooth approximate functional equation for $L(s, \chi_d)$, which is representable in terms of the incomplete gamma function. In the latter case, we consider the Dedekind zeta function for the associated quadratic field, reducing the computation of $L(1/2, \chi_d)$ to a sum over binary quadratic forms, and the K-Bessel expansions of their Epstein zeta functions as determined in [Chowla and Selberg 67].

Testing the conjectures described in the introduction also involves numerical values for the coefficients of the polynomials $Q_{\pm}(k, x)$. We reran the program used in [Conrey et al. 05] on a faster machine and for a longer amount of time to get slightly more accurate coefficients for these polynomials.

3.1. Computational Formula for $L(1/2, \chi_d)$, $d < 0$

Let

$$\zeta_{\mathbb{Q}(\sqrt{D})}(s) = \zeta(s)\, L(s, \chi_d)$$

be the Dedekind zeta function of the quadratic number field $\mathbb{Q}(\sqrt{D})$, and $h(d)$ the corresponding class number.

Let $a_j m^2 + b_j mn + c_j n^2$, $j = 1, \dots, h(d)$, be representatives for the $h(d)$ equivalence classes of primitive positive definite binary quadratic forms of discriminant $b_j^2 - 4a_j c_j = d < 0$ [Landau 66].

Dirichlet proved (see also [Davenport 00, Chapter 6]) that

$$\zeta_{\mathbb{Q}(\sqrt{D})}(s) = \frac{1}{\omega}\sum_{j=1}^{h(d)} {\sum}' \left(a_j m^2 + b_j mn + c_j n^2\right)^{-s}, \quad \Re s > 1, \tag{3–1}$$

where

$$\omega = \begin{cases} 2, & \text{if } d < -4, \\ 4, & \text{if } d = -4, \\ 6, & \text{if } d = -3, \end{cases}$$

and where $\sum'$ denotes the sum over all pairs $(m, n) \in \mathbb{Z}^2$, $(m, n) \ne (0, 0)$.

Chowla and Selberg obtained the meromorphic continuation of the Epstein zeta function

$$Z(s) := {\sum}' (am^2 + bmn + cn^2)^{-s}, \quad \Re s > 1,$$

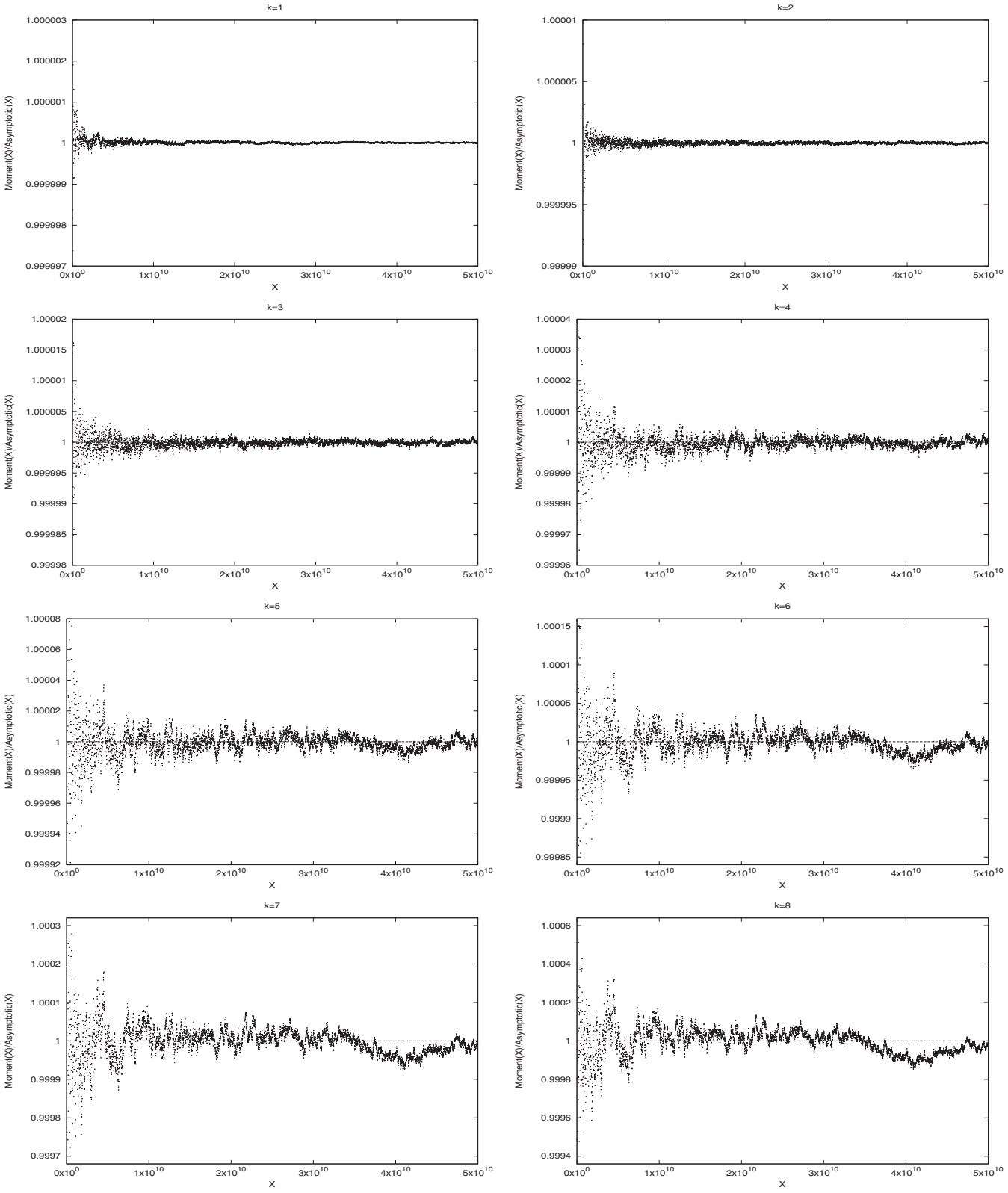

FIGURE 1. These plots depict the ratio $R_{-}(k, X)$ of the numerically computed moments compared to the CFKRS predictions for $k = 1, \ldots, 8$ and $d < 0$, sampled every 10^7, i.e., at $X = 10^7, 2 \times 10^7, \ldots, 5 \times 10^{10}$. The horizontal axis is X, the vertical axis is the ratio $R_{-}(k, X)$.

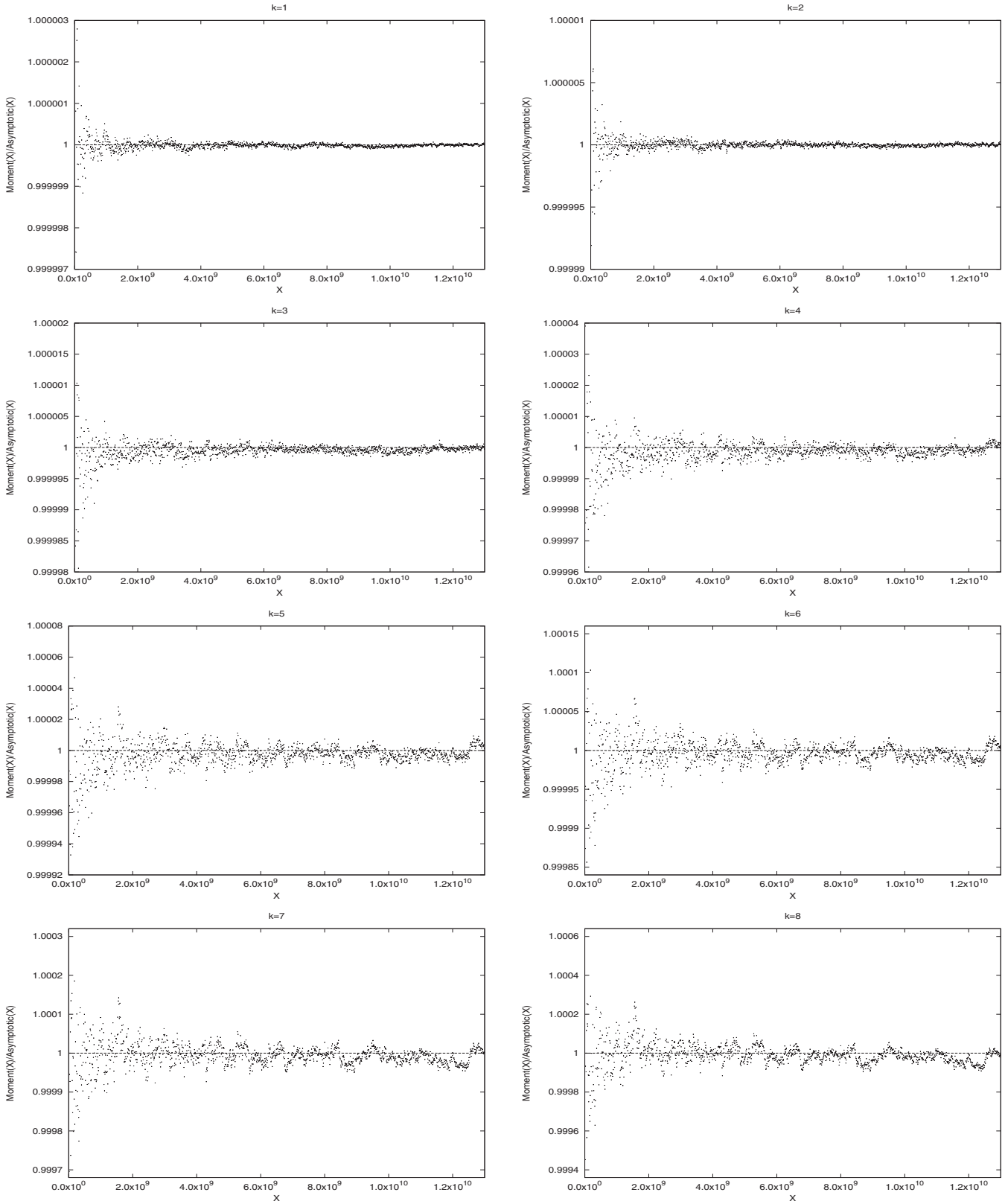

FIGURE 2. These plots depict the ratio $R_+(k, X)$ of the numerically computed moments compared to the CFKRS predictions for $k = 1, \ldots, 8$ and $d > 0$, sampled every 10^7, i.e., at $X = 10^7, 2 \times 10^7, \ldots, 1.3 \times 10^{10}$. The horizontal axis is X, the vertical axis is the ratio $R_+(k, X)$.

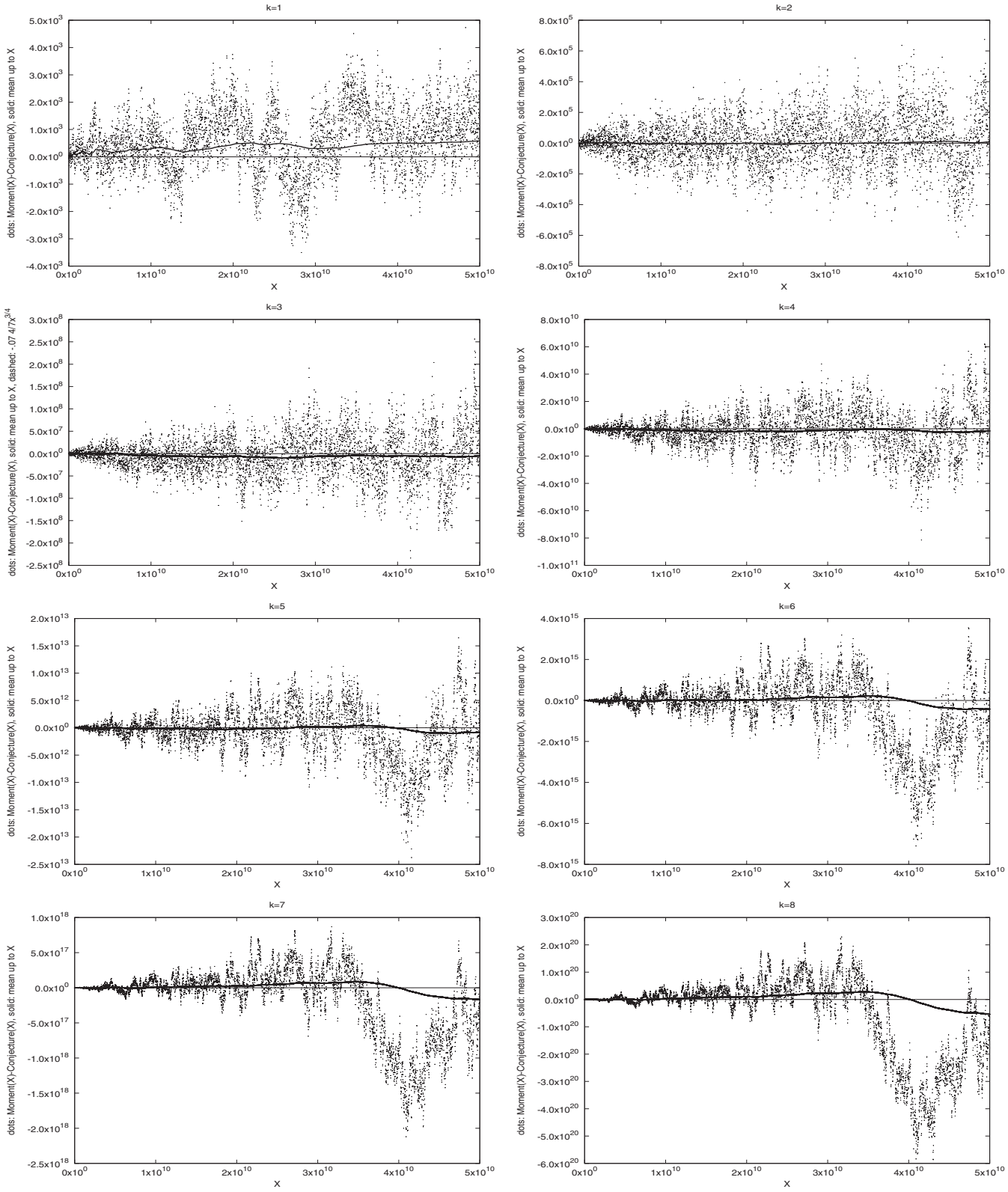

FIGURE 3. These plots depict the difference $\Delta_{-}(k, X)$ between the numerically computed moments and the CFKRS prediction, for $k = 1, \ldots, 8$ and $d < 0$, sampled at $X = 10^7, 2 \times 10^7, \ldots, 5 \times 10^{10}$. The horizontal axis is X, the vertical axis is the difference $\Delta_{-}(k, X)$, and the solid curve is the mean up to X of the plotted differences (see the discussion in the text).

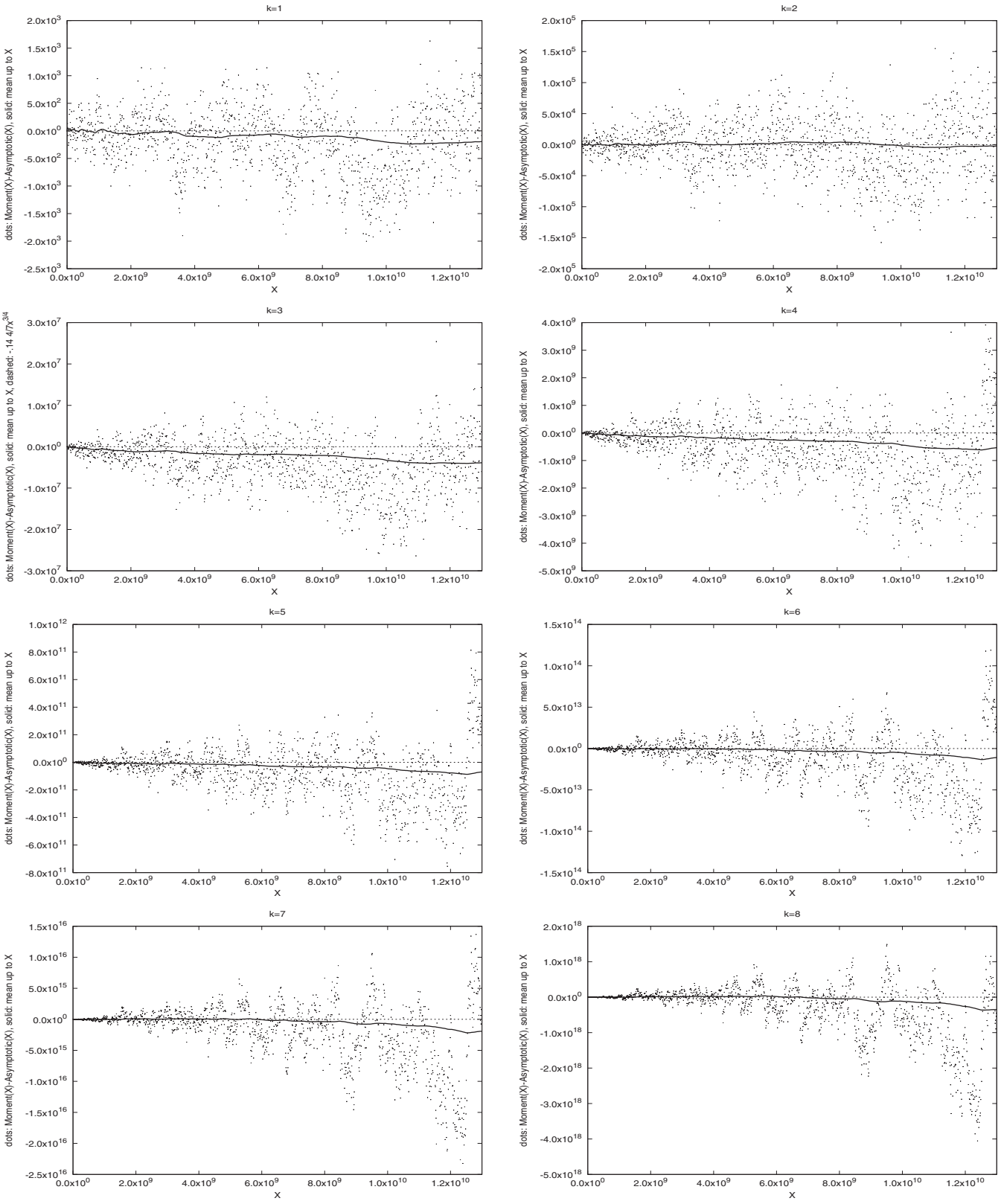

FIGURE 4. These plots depict the difference $\Delta_+(k, X)$ between the numerically compute moments and the CFKRS prediction for $k = 1, \ldots, 8$ and $d > 0$, sampled at $X = 10^7, 2 \times 10^7, \ldots, 1.3 \times 10^{10}$. The horizontal axis is X, the vertical axis is the difference $\Delta_+(k, X)$, and the solid curve is the mean up to X of the plotted differences (see the discussion in the text).

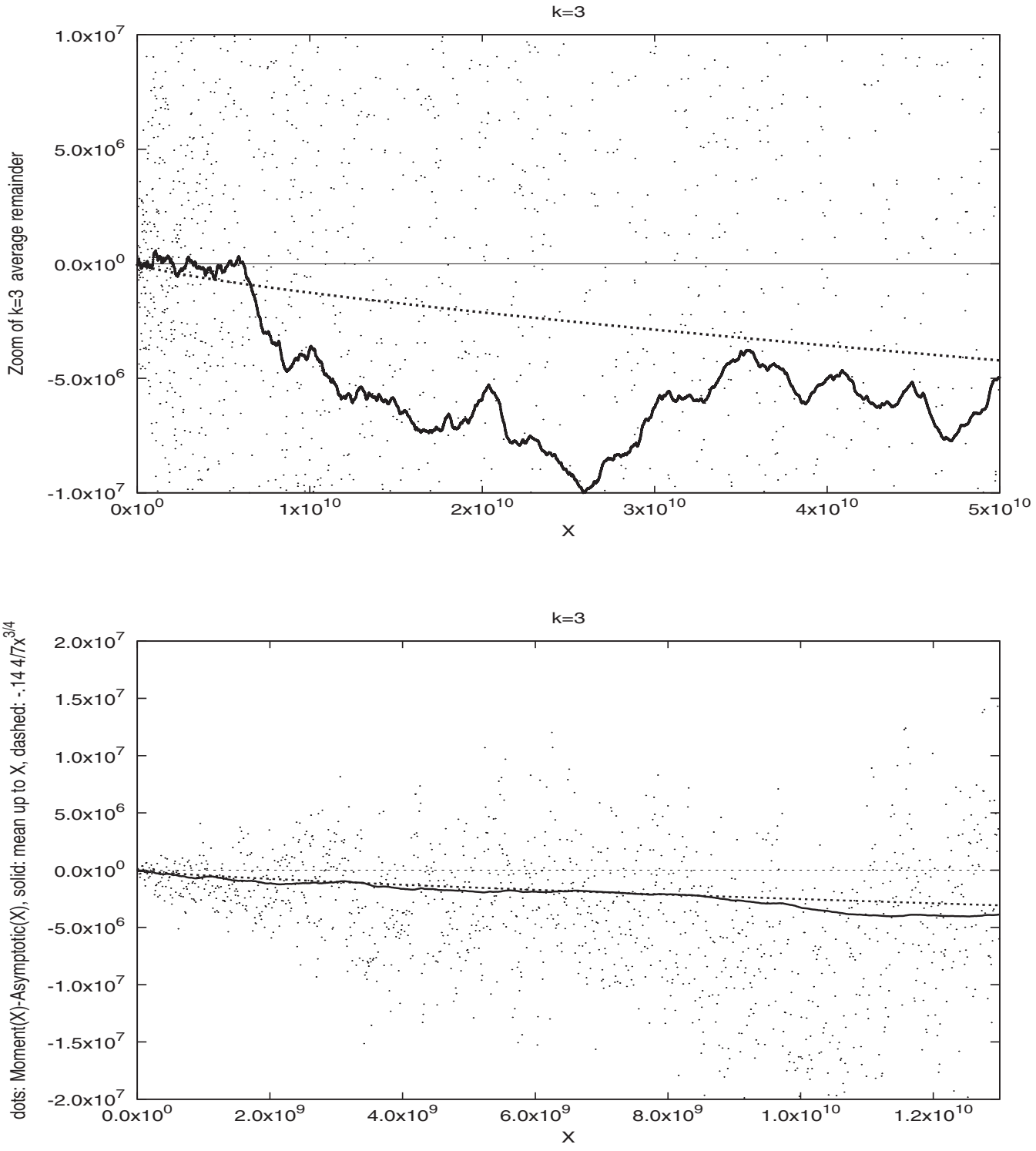

FIGURE 5. This graph depicts the remainder term for the third moment for $d < 0$ (top) and $d > 0$ (bottom), i.e., $\Delta_{-}(3, X)$ and $\Delta_{+}(3, X)$. The solid line is the average remainder, and the dashed line is Zhang's prediction. See the discussion in the text concerning the quality of the fit.

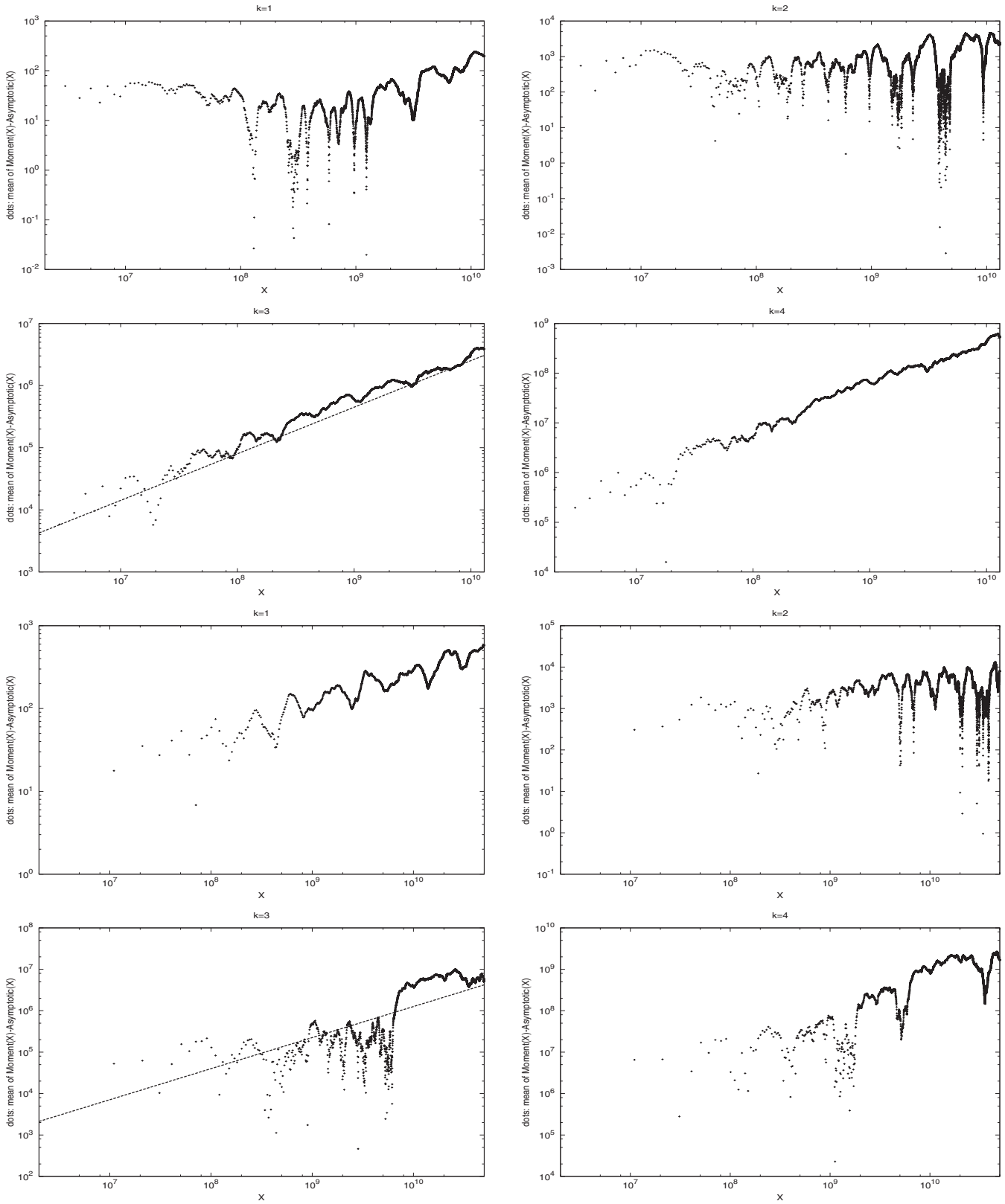

FIGURE 6. Plots, on a log log scale, of the absolute value of the average remainder term depicted in Figures 3 and 4, for $1 \leq k \leq 4$, $d > 0$ (top four plots), and $d < 0$ (bottom four plots). For the third moment we compare to Zhang's predictions of $0.14\frac{4}{7}x^{3/4}$ (third plot) and $0.07\frac{4}{7}x^{3/4}$ (seventh plot). For the first moment, $d < 0$, there seems to be a bias of size roughly $x^{1/4}$.

with $d = b^2 - 4ac < 0$, $a, c > 0$, by giving an expansion for $Z(s)$ as a series of K-Bessel functions [Chowla and Selberg 67]. Specifically, they proved that

$$Z(s) = 2\zeta(2s)a^{-s} + \frac{2a^{s-1}\sqrt{\pi}}{\Gamma(s)\left(|d|^{1/2}/2\right)^{2s-1}}\zeta(2s-1) \times \Gamma\left(s - \frac{1}{2}\right) + B(s), \tag{3–2}$$

where

$$B(s) = \frac{8\pi^s 2^{s-1/2}}{a^{1/2}\Gamma(s)\ |d|^{(2s-1)/4}} \times \sum_{n=1}^{\infty} n^{s-1/2}\ \sigma_{1-2s}(n)\cos\left(\frac{n\pi b}{a}\right) K_{s-1/2}\left(\frac{\pi n\ |d|^{1/2}}{a}\right),$$

$$\sigma_\omega(n) = \sum_{m|n} m^\omega,$$

and

$$K_\omega(z) = \frac{1}{2}\int_0^\infty \exp\left(-\frac{z}{2}\left(y + \frac{1}{y}\right)\right) y^{\omega-1}dy, \quad \Re z > 0.$$

The function $K_{s-1/2}(x)$ decreases exponentially fast as $x \to \infty$, uniformly for s in compact sets, and the above expansion gives $Z(s)$ as an analytic function throughout $\mathbb{C}$ except for a simple pole at $s = 1$. Note that the poles at $s = 1/2$ in (3–2) of the terms with $\zeta(2s)$ and $\Gamma(s - 1/2)$ cancel out.

Specializing the above formula to $s = 1/2$ gives

$$Z\left(\frac{1}{2}\right) = \frac{2}{a^{1/2}}\left(\gamma + \log\left(\frac{|d|^{1/2}}{8\pi a}\right)\right) + \frac{8}{a^{1/2}}\sum_{n\geq 1}\sigma_0(n)\cos\left(\frac{\pi nb}{a}\right) K_0\left(\frac{\pi n\,|d|^{1/2}}{a}\right). \tag{3–3}$$

Substituting this into (3–1) for a given set of representative quadratic forms, one for each equivalence class, yields a formula that can be used to numerically compute $L(1/2, \chi_d)$.

An explicit bound on $K_0(x)$ can be obtained as follows:

$$K_0(x) = \int_1^\infty \exp\left(-\frac{x}{2}\left(y + \frac{1}{y}\right)\right)\frac{dy}{y},$$

so that

$$|K_0(x)| < \sqrt{\frac{\pi}{2x}}e^{-x}. \tag{3–4}$$

The last inequality can be verified by writing

$$y + 1/y = \left(y^{1/2} - y^{-1/2}\right)^2 + 2,$$

changing variables

$$u = x^{1/2}\left(y^{1/2} - y^{-1/2}\right),$$

so that

$$\frac{dy}{y} = x^{-1/2}2\frac{du}{y}^{1/2} + y^{-1/2},$$

and using, from the arithmetic–geometric mean inequality, $y^{1/2} + y^{-1/2} \geq 2$.

Next, we discuss implementation issues and complexity arising from this formula.

Lagrange proved that each of the equivalence classes of primitive positive definite forms of discriminant d contains exactly one form $ax^2 + bxy + cy^2$ for which $-a < b \leq a < c$ or $0 \leq b \leq a = c$. Roughly, this is the set $0 \leq |b| \leq a \leq c$, with some exceptions. Furthermore, $a < (|d|/3)^{1/2}$. Recall that in this context, primitive means that $\gcd(a, b, c) = 1$.

Therefore, let $A(X)$ denote the set of triples

$$A(X) := \left\{(a, b, c) \in \mathbb{Z}^3 \mid 4ac - b^2 \leq X,\ -a < b \leq a < c \text{ or } 0 \leq b \leq a = c\right\} \tag{3–5}$$

and $A'(X)$ the set of primitive triples in $A(X)$:

$$A'(X) := \{(a, b, c) \in A(X) \mid \gcd(a, b, c) = 1\}. \tag{3–6}$$

Our first step was to distribute the computation across several processors, each one handling a range of discriminants, in order to speed up the computation and also to reduce the memory requirements per processor. Therefore, suppose $0 < -d \leq X$ for some positive integer X, and let ΔX be a positive integer dividing X. We partitioned the interval into blocks, $X_{i-1} < |d| \leq X_i$, of equal length $\Delta X = X_i - X_{i-1}$ for $i = 1, 2, \ldots, m$ (so $X_m := X$).

We then looped through all integers $(a, b, c) \in A'(X)$, with corresponding $|d|$ lying in $(X_{i-1}, X_i]$, i.e., satisfying the following properties:

$$0 < a \leq \sqrt{\frac{X_i}{3}}, \quad 0 \leq |b| \leq a \leq c, \tag{3–7}$$
$$\frac{b^2 + X_{i-1}}{4a} < c \leq \frac{b^2 + X_i}{4a}$$

(taking care to throw away the terms above at the endpoints that are not in $A'(X)$, for example, terms with $-b = a$).

We computed $d = b^2 - 4ac$ and updated $L(1/2, \chi_d)$, stored in an array, using (3–3) to calculate the corresponding contribution to (3–1) from the triple (a, b, c).

Combining $a < (|d|/3)^{1/2}$ with the exponential decay of $K_0(x)$ in (3–4) shows that few terms are needed to compute (3–3) to given precision. For example, the terms in (3–3) with $n \geq 7$ contribute, in absolute value, less than 10^{-15} to the sum, and can be ignored. Smaller a require even fewer terms.

Remark 3.1. Because we are evaluating $K_0(x)$ for a limited range of values and to machine double precision (15 or 16 digits), we used a precomputed table of the first five terms of several thousand Taylor series expansions,

$$K_0(x) \approx K_0(x_j) + K_0^{(1)}(x_j)(x - x_j) + \cdots + \left(\frac{K_0^{(4)}(x_j)}{4!}\right)(x - x_j)^4, \tag{3–8}$$

each one centered on a point x_j of the form $x_j = j/200$ with $5 < x_j < 37$, $j \in \mathbb{Z}$. The interval $[5, 37]$ was used, because on one end, we have $\pi 3^{1/2} > 5$, the left-hand side being a lower bound for the smallest possible x for which we would need to evaluate $K_0(x)$. We chose 37 for the other end of the interval because $\exp(-37) < 10^{-16}$, i.e., smaller than our desired precision.

While our code was written in C++, the precomputation of these Taylor expansions was carried out in Maple, with the coefficients stored in a file that could be read into our C++ program. Our C++ code was compiled with the GNU compiler `GCC`.

Only five terms were computed and stored, because our Taylor expansions were always applied with $|x - x_j| < 1/400$. Combined with the exponential decay of the Taylor coefficients (as a function of x), at most five terms, and often fewer, were needed to evaluate the sum to within 10^{-15}.

We also make note of a few of the hacks that helped to increase the speed of our program:

1. For a given a, b, only one cosine needs to be computed. Indeed, given $\cos(\pi b/a)$, we can compute $\cos(\pi n b/a)$, for $n = 1, 2, \ldots, 7$, using standard trigonometric identities. For instance, the double-angle identity computes the expression for $n = 2$.

2. To test for primitivity, we must check whether $\gcd(a, b, c) = 1$. If one computes $\gcd(a, b)$ outside the c loop as previously mentioned, then for a given $\gcd(a, b)$, we can use

$$\gcd(a, b, c) = \gcd(\gcd(a, b), c \bmod \gcd(a, b)),$$

and thus compute, and then store within the c loop, at most one gcd per residue class modulo $\gcd(a, b)$.

3. When reading an array, the computer loads blocks of consecutive bytes of the array from RAM into the CPU's cache, where it can be accessed quickly by the CPU. On profiling, we found, after optimizing and streamlining the bulk of our code, that a significant amount of time was being spent accessing the array in which we were storing $L(1/2, \chi_d)$ so as to increment it by the contribution from a given triple (a, b, c). The reason was that as the inner c loop increments by 1, the value of $d = b^2 - 4ac$ changes by $4a$, i.e., often a large step. Nonsequential array accesses are expensive with respect to time. We were able to significantly decrease the time spent on array accesses by anticipating the subsequent d's, and using `GCC`'s `__builtin_prefetch` function to fetch the corresponding array entry for $L(1/2, \chi_d)$ eight turns in advance; the eight was determined experimentally on the hardware that we used and for the range of d's that we considered.

4. The computation of (3–3) involves $\log(|d|)$. Therefore, we precomputed these values and stored them in an array, but were again faced with the same kind of expensive memory accesses as for the values of $L(1/2, \chi_d)$. Rather than prefetch these separately, we created a C++ struct to hold both $L(1/2, \chi_d)$ and $\log(|d|)$ together. In that way, a single prefetch would load both at once.

Remark 3.2. On combining the last two hacks, the array access portion of our code sped up by a factor of 4, and the overall running time of the program sped up by a factor of 2. These two hacks were the last two implemented, and the speedup achieved indicates how expensive nonsequential memory accesses can be, and how optimized the rest of our code was.

5. To avoid repeatedly checking whether the quantity $d = b^2 - 4ac$ was a fundamental discriminant, we precomputed, for each block $X_{i-1} < |d| \le X_i$, whether d is a fundamental discriminant and stored that information in an array of Boolean variables. We essentially sieved for square-free numbers, and this can be done, for each block of length ΔX, in $O(\Delta X)$ steps, because the sum of the reciprocal of the squares converges. Hence, doing so across all blocks up to X costs $O(X)$ arithmetic operations and array accesses. No prefetching was used on this array, since it did not seem to give a benefit, perhaps because the array consists of single-bit Boolean variables, rather than 64-bit doubles of the L-value and $\log(|d|)$ arrays, and fits more easily within cache.

6. Since $\cos(x)$ is an even function and b gets squared in the discriminant equation $|d| = 4ac - b^2$, we can group $\pm b$ together, when possible, and restrict our attention to nonnegative b values. Only a relatively small subset of triples cannot be paired in this fashion, namely when $a = b$, $b = 0$, or $c = a$.

7. Terms such as

$$\frac{2}{a^{1/2}}(\gamma - \log(8\pi a))$$

appearing in the leading term of (3–3) depend solely on a. As such, it is to our advantage to compute this, and all other terms depending solely on a, outside the b and c loops. Similarly, we compute expressions like $\gcd(a, b)$ outside the c loop, and so on. While this is standard, we took it to a meticulous extreme to save as many arithmetic operations as possible.

3.2. Complexity for $d < 0$

First observe that the number of candidate triples (a, b, c) defined by (3–5) that we must loop over satisfies

$$|A(X)| \sim \frac{\pi}{18} X^{3/2}. \tag{3–9}$$

See [Kuhleitner 02] for a discussion of this counting problem. Furthermore, the number of triples that survive the condition $\gcd(a, b, c) = 1$ is

$$|A'(X)| \sim \frac{\pi}{18\zeta(3)} X^{3/2}. \tag{3–10}$$

The latter asymptotic formula was essentially stated by Gauss in his *Disquisitiones* in connection with the sum of class numbers $A'(X) = \sum_{-X<d<0} h(d)$. A proof, with a lower-order term and a bound on the remainder, as well as further references can be found in [Chamizo and Iwaniec 98].

We can get a lower bound for the amount of computation required by simply counting the number of triples (a, b, c) that are considered. Furthermore, because we are pairing together $\pm b$, the number of triples that survive the gcd condition is roughly half of (3–10), i.e.,

$$\sim \frac{\pi}{36\zeta(3)} X^{3/2}. \tag{3–11}$$

The relatively small constant $\pi/36\zeta(3)$ helps to make this approach very practical.

While the above asymptotic gives a lower bound on the number of operations of our method in computing all $L(1/2, \chi_d)$ for $0 < -d \le X$, it ignores the amount of work needed for each triple. Most of the work involves checking bounds on each loop, testing whether $\gcd(a, b, c) = 1$ and whether $d = b^2 - 4ac$ is a fundamental discriminant, and carrying out simple arithmetic and array accesses related to the evaluation of (3–3).

Each arithmetic operation can be done in polynomial time in the size of the numbers involved (in fact, $\log(X)^{1+\epsilon}$ using the FFT). However, in the range of discriminants we considered ($|d| < 5 \times 10^{10}$), the bit length is quite small: 32-bit C++ `int`s for a, b, c sufficed and 64-bit `long int`s were used for discriminants. We also used 64-bit machine doubles for the floating-point arithmetic. Therefore, all arithmetic was carried out in hardware.

Recall that we are assuming that $0 < -d \le X$, and partitioning this interval as

$$\underbrace{1, \ldots, \Delta X}_{\text{Block 1}}, \underbrace{\Delta X + 1, \ldots, 2\Delta X}_{\text{Block 2}}, \ldots,$$
$$\underbrace{(m-1)\Delta X + 1, \ldots, m\Delta X}_{\text{Block m}}, \ldots,$$

where ΔX is a positive integer assumed to divide X (in practice, one can take X and ΔX to be powers of, say, ten). The number of blocks equals $X/\Delta X$, where, for reasons made clear below, we will eventually take $\Delta X \gg X^{1/2} \log(X)^2$.

As mentioned earlier, we precomputed, via sieving, a table of fundamental discriminants for each block of length ΔX using $O(\Delta X)$ arithmetic operations and array accesses, hence $O(X)$ operations across all blocks, i.e., a relatively small cost compared to $O(X^{3/2})$.

The number of terms needed in the expansion of the Bessel function K_0 is proportional to the number of digits of precision desired, and the number of Taylor coefficients used in each Taylor series for $K_0(x)$ also depends on the output precision. We worked with machine doubles and hence to a precision of about 15 to 16 digits, and as explained in Remark 3.1, we used at most five terms in each Taylor series.

Next we consider the time used to carry out the gcd computations. We described in the hacks listed above that for each triple (a, b, c), we computed $\gcd(a, b, c)$ by first computing $\gcd(a, b)$ outside the c-loop and then computing $\gcd(\gcd(a, b), c \bmod \gcd(a, b))$ inside the c-loop, with the latter calculation being performed at most once per residue class modulo $\gcd(a, b)$, and then stored in the c-loop for subsequent use.

The number of $\gcd(a, b)$'s that are computed across all blocks is

$$\ll X \sum_{m \le \frac{X}{\Delta X}} \sum_{a \le \sqrt{\frac{m\,\Delta X}{3}}} \sum_{0 \le b \le a} 1 \ll \frac{X^2}{\Delta X}. \tag{3–12}$$

Next, we compute

$$\gcd(a, b, c) = \gcd(\gcd(a, b), c \mod \gcd(a, b)),$$

storing these values, inside the a, b loops, for each residue class modulo $\gcd(a, b)$ so as to compute only one gcd per residue class. Therefore, the number of additional gcd's required is

$$\ll \sum_{m \le \frac{X}{\Delta X}} \sum_{a \le \sqrt{\frac{m\,\Delta X}{3}}} \sum_{0 \le b \le a} \gcd(a, b). \tag{3–13}$$

It is known that

$$\sum_{a\le x}\sum_{0\le b\le a}\gcd(a,b)\sim\frac{x^2\log x}{2\zeta(2)};$$

see, for example, [Bordellès 07] or [Broughan 01]. So, from (3–13) and the above asymptotic, we see that the number of additional gcd computations required across all blocks is

$$\ll\sum_{m\le\frac{X}{\Delta X}} m\Delta X\log(m\Delta X)\ll\frac{X^2\log X}{\Delta X}.$$

Therefore, combining this with (3–12), the total number of gcd calls is

$$O\left(\frac{X^2\log X}{\Delta X}\right).$$

Furthermore, each $\gcd(a,b)$ can be computed, using the Euclidean algorithm, in $O\big(\log(X)^2\big)$ bit operations, since the binary length of both a and b is $O(\log X)$, and so the total number of bit operations coming from gcd's is $\ll X^2\log(X)^3/\Delta X$.

Hence, we can make the overall time required for gcd evaluations an insignificant portion of the overall time by choosing

$$\Delta X\gg X^{1/2}\log(X)^2, \tag{3–14}$$

since the number of bit operations for the remaining work (looping through a, b, c, and m, and carrying out the required integer and floating-point arithmetic) is then

$$O\left(X^{3/2}\log(X)^{1+\epsilon}\right).$$

The $X^{3/2}$ accounts for the overall number of triples (a,b,c) considered, and the $\log(X)^{1+\epsilon}$ for the cost of arithmetic on numbers of bit length $O(\log X)$. The implied constant depends on the number of digits of precision desired for the L-values.

While the best choice might seem to be to take ΔX equal to X so as to minimize the number of gcd calls, this would come at a substantial price. First, such a large ΔX would prevent us from simply distributing the computation across several processors, each one handling one block at a time.

Second, the memory (RAM) requirements needed would be enormous. There is also an advantage to having arrays that can fit entirely or significantly within the CPU's cache, so as to avoid too many expensive memory fetches from RAM, and even with smaller ΔX, there is a tradeoff between minimizing calls to the Euclidean algorithm and memory accesses.

We determined a good choice of ΔX experimentally, since in practice, the big-O constants in the above estimates depend on the speed of individual arithmetic and memory operations on given hardware and context in which they are called.

Nonetheless, since the Euclidean algorithm is very simple, and the remaining work associated with looping, computing the K-Bessel function, and updating L-values involves a moderate number of arithmetic and memory operations, one expects that the benefit should be felt sooner rather than later. Indeed, we found that in our range of d's, a choice that eliminated the gcd's as a bottleneck while not paying too high of a cache size penalty was $\Delta X=10^6$, i.e., block sizes of one million.

3.3. Computational Formula for $L(1/2, \chi_d)$, $d > 0$

De la Vallée Poussin's proof of the functional equation for $L(s,\chi_d)$ imitates that of Riemann for his zeta function. It yields the analytic continuation of $L(s,\chi_d)$ and also the following formula, an example of a "smoothed approximate functional equation," useful for its evaluation:

$$\left(\frac{d}{\pi}\right)^{s/2}\Gamma\left(\frac{s}{2}\right)L(s,\chi_d)$$
$$=\sum_{n=1}^{\infty}\chi_d(n)\left(G\left(\frac{s}{2},\frac{\pi n^2}{d}\right)+G\left(\frac{1-s}{2},\frac{\pi n^2}{d}\right)\right),$$

where $G(z,w)$ denotes the normalized incomplete gamma function

$$G(z,w):=\int_1^{\infty}x^{z-1}e^{-wx}dx$$
$$=w^{-z}\int_w^{\infty}x^{z-1}e^{-x}dx=w^{-z}\Gamma(z,w),\quad \Re w>0,$$

with $\Gamma(z,w)$ the incomplete gamma function. See, for instance, [Davenport 00, p. 69], or [Rubinstein 05, Section 3.4.1], and use Gauss's formula for the Gauss sum, namely $\tau(\chi_d)=d^{1/2}$, when $d>0$.

Therefore, on specializing to $s=1/2$, we have a smooth approximate functional equation for $L(1/2,\chi_d)$, namely

$$L\left(\frac{1}{2},\chi_d\right)=2\Big(\frac{\pi}{d}\Big)^{1/4}\sum_{n\ge1}\chi_d(n)\frac{G(1/4,n^2\pi/d)}{\Gamma(1/4)}$$
$$=2\sum_{n\ge1}\frac{\chi_d(n)}{\sqrt{n}}\frac{\Gamma(1/4,n^2\pi/d)}{\Gamma(1/4)}, \tag{3–15}$$

valid for positive fundamental discriminants d.

To estimate the size of the terms being summed, first notice, from the definition, that $G(1/4,w)>0$ for real w.

Furthermore, integrating by parts gives an upper bound:

$$G\left(\frac{1}{4}, w\right) = \frac{e^{-w}}{w} - \frac{3}{4w}\int_1^\infty e^{-wx}x^{-7/4}dx < \frac{e^{-w}}{w}. \tag{3–16}$$

This inequality tells us that the terms in (3–15) decrease exponentially fast in the quantity $\pi n^2/d$, so that roughly speaking, we need to truncate the sum when n is of size $d^{1/2}$ to achieve a small tail.

Let us consider this estimate more carefully. Set

$$f(t) = \frac{2}{\sqrt{t}}\frac{\Gamma\big(1/4, t^2\pi/d\big)}{\Gamma(1/4)}. \tag{3–17}$$

In light of bound (3–16), we have

$$|f(n)| < \frac{2}{\Gamma(1/4)}\left(\frac{d}{\pi}\right)^{3/4}\frac{e^{-\pi n^2/d}}{n^2}. \tag{3–18}$$

Let the number of working digits be labeled digits. Hence, for

$$n > \sqrt{\frac{d}{\pi}\log(10)\cdot \mathsf{digits}}, \tag{3–19}$$

we generously have

$$f(n) < 10^{-\mathsf{digits}}.$$

Furthermore, notice that the terms start off, for smaller n and large d, with

$$\frac{\Gamma(1/4, n^2\pi/d)}{\Gamma(1/4)} \sim 1.$$

Therefore, it does not make sense to sum the terms beyond (3–19), since those terms are lost to numerical imprecision.

We must thus see to what extent the ignored tail end of the sum can contribute to the value of $L(1/2, \chi_d)$. Summation by parts yields

$$\sum_{n\le N}\chi_d(n)f(n) = f(N)\sum_{n\le N}\chi_d(n) - \int_1^N \sum_{n\le t}\chi_d(n)f'(t)dt. \tag{3–20}$$

So on letting $N\to\infty$, we obtain

$$L(1/2,\chi_d) = -\int_1^\infty \sum_{n\le t}\chi_d(n)f'(t)dt. \tag{3–21}$$

Moreover, by subtracting (3–20) from (3–21), we get a formula for the tail:

$$\sum_{n=N+1}^{\infty}\chi_d(n)f(n) \tag{3–22}$$
$$= -f(N)\sum_{n\le N}\chi_d(n) - \int_N^\infty\sum_{n\le t}\chi_d(n)f'(t)dt.$$

One could use the inequality of Polyá–Vinogradov, that of Burgess, or even the trivial bound $|\chi_d(n)|\le 1$ here to get a reasonable, but not optimal, estimate for the size of the tail. However, something closer to the truth is obtained by using the conjectured bound

$$\sum_{n\le x}\chi_d(n) = O\Big(x^{1/2}d^\epsilon\Big). \tag{3–23}$$

Combined with (3–18), this gives

$$f(N)\sum_{n\le N}\chi_d(n) = O\left(\frac{d^{3/4+\epsilon}}{N^{3/2}}e^{-\pi N^2/d}\right),$$

and similarly,

$$\int_N^\infty\sum_{n\le t}\chi_d(n)f'(t)dt \ll d^\epsilon\int_N^\infty t^{\frac{1}{2}}f'(t)dt \ll \frac{d^{3/4+\epsilon}}{N^{3/2}}e^{-\pi N^2/d},$$

where we have applied integration by parts to get the last bound. Applying these bounds to (3–22) and choosing

$$N = \sqrt{\frac{d}{\pi}\log(10)\cdot\mathsf{digits}} \tag{3–24}$$

gives the following bound for (3–22):

$$O\left(10^{-\mathsf{digits}}\frac{d^\epsilon}{\mathsf{digits}^{1/2}}\right). \tag{3–25}$$

We therefore conclude that the tail is not much bigger than an individual term, and in principle, we could compensate for the extra d^ϵ by taking digits slightly larger than our desired output precision, say by an amount equal to $\epsilon\log d/\log 10$.

We remark that by using the trivial estimate or Polyá–Vinogradov inequality, we could get rigorous estimates with explicit constants, but larger by a factor of roughly $d^{1/4}$ compared to (3–25).

3.4. Cancellation and Accuracy

We can also use the above analysis to show that our approach to computing $L(1/2,\chi_d)$ using the smooth approximate functional equation is well balanced, i.e., that little cancellation and hence loss of precision takes place in summing (3–15). We consider the maximum size that the partial sums can attain to give us a sense of how many digits of accuracy after the decimal place are attained in working with digits decimal places.

Consider the partial sums (3–20) (for a general N', not just our specific choice of N), apply the conjectured

bound (3–23), and integrate by parts:

$$\sum_{n\le N'} \chi_d(n) f(n) \ll f(N') N'^{1/2} d^\epsilon + d^\epsilon t^{1/2} f(t)\Big|_1^{N'} + d^\epsilon \int_1^{N'} t^{-1/2} f(t)\,dt. \tag{3–26}$$

Again, we can get a proven, though weaker, upper bound with explicit constants if we use a proven bound rather than the conjecture (3–23).

Next, notice that $\Gamma(1/4, x) < \Gamma(1/4)$, because the definition of the left-hand side here involves integrating over a smaller portion of the positive real axis compared to the right-hand side. Thus, from (3–17), we have

$$f(t) < 2t^{-1/2}.$$

Applying this to (3–26) gives

$$\sum_{n\le N'} \chi_d(n) f(n) \ll d^\epsilon \log N'.$$

Therefore, because we take the partial sums with $N' \le N = O((d\,\mathsf{digits})^{1/2})$, we have, on adjusting ϵ to incorporate $\log d$:

$$\sum_{n\le N'} \chi_d(n) f(n) \ll d^\epsilon \log \mathsf{digits}.$$

Therefore, the partial sums do not get large, and we thus have nearly as many digits of accuracy beyond the decimal place as our working precision.

Using a similar analysis, the effect of accumulated round-off error can be estimated by replacing $\chi_d(n)$ with random plus and minus 1's multiplied by a factor of size $10^{-\mathsf{digits}}$ to model the random rounding up or down of the terms in the sum. With high probability, we then get an error, due to accumulated round-off of size

$$(d^\epsilon \log \mathsf{digits}) 10^{-\mathsf{digits}}.$$

If one desires rigorous, rather than experimental, values of $L(1/2, \chi_d)$, an interval arithmetic package should be used in practice. Because our goal was to test conjectures rather than prove a rigorous numerical result, we were satisfied with an intuitive understanding of the accuracy of our computation, and we carried out several checks of the values attained, for example comparing a similar smooth approximate functional equation for the case $d < 0$ against select values attained by our implementation using the Epstein zeta function, and also using a high-precision version of Rubinstein's `lcalc` package to test several values.

3.4.1. Hacks

We list a few hacks that were helpful in the implementation of the smooth approximate functional equation (3–15).

1. $\chi_d(n)$ can be efficiently computed by repeatedly extracting powers of 2 and applying quadratic reciprocity.

2. As in the case for $d < 0$, it is to our advantage to partition $0 < d \le X$ into blocks and farm the work out to many processors.

3. Due to the presence of $\chi_d(n)$ in (3–15), it is more efficient to place the d-loop on the inside and the n-loop on the outside, because $\chi_d(n)$ is periodic in d with period either n or $8n$, depending on whether the power of two dividing n is even or odd. Furthermore, n is comparatively small compared to d, by (3–24). Thus, for each n we precomputed a table of $\chi_d(n)$, so as to compute this value only once per residue class d modulo n or $8n$. This pays off as long as each residue class gets hit, on average, more than once (perhaps slightly more because of the overhead involved in storing the values and looking up the array). In our implementation, with blocks of length 10^6, $0 < d < 1.3 \times 10^{10}$, and 16 digits of working precision, it was useful to do so.

4. We computed the normalized incomplete gamma function $G(z, w)$, evaluated at $z = 1/4$ and $w = n^2\pi/d$, as follows. For $w > 37$, return 0 (since $\exp(-37) < 10^{-16}$). For $1 < w < 37$, use a precomputed table of Taylor series, centering each Taylor series at multiples of 0.01 (so nearly 4000 Taylor series) and taking terms up to degree 7 (less for larger w because of the exponential decay). Otherwise, for $w < 1$, employ the complementary incomplete gamma function

$$\gamma(z, w) := \Gamma(z) - \Gamma(z, w) = \int_0^w e^{-x} x^{z-1}\,dx,$$

$\Re(z) > 0$, $|\arg| w < \pi$. Specifically, set

$$g(z, w) = w^{-z}\gamma(z, w) = \int_0^1 e^{-wt} t^{z-1}\,dt,$$

so $G(z, w) = w^{-z}\Gamma(z) - g(z, w)$, and integrate by parts to get

$$g(z, w) = e^{-w} \sum_{j=0}^{\infty} \frac{w^j}{(z)_{j+1}},$$

where

$$(z)_j = \begin{cases} z(z+1)\cdots(z+j-1) & \text{if } j > 0, \\ 1 & \text{if } j = 0. \end{cases}$$

We stored the value of $\Gamma(1/4)$ and calculated the above series for $g(1/4, w)$ by truncating the sum when the tail was less than 10^{-16}.

3.5. Complexity for $d > 0$

Recall that as for the case of negative discriminants, we are partitioning the interval $0 < d < X$ into blocks of length ΔX.

The overall cost for sieving for fundamental discriminants, summed over all blocks, is a meager $O(X)$ arithmetic operations and array accesses on numbers of bit length $O(\log X)$, as for the case of $d < 0$.

Next we estimate the overall time, summed over blocks, required to create a precomputed table of characters $\chi_d(n)$ for all residue classes modulo n or $8n$.

Summing over blocks m and taking the maximum truncation point (3–24) that occurs for a given block, the time required is

$$\ll \log(X)^2 \sum_{m \le X/\Delta X} \sum_{n \le M} n,$$

where

$$M = \sqrt{\frac{m\Delta X}{\pi} \log(10) \cdot \mathsf{digits}}\,. \tag{3–27}$$

Here we have used the fact that each character $\left(\frac{d}{n}\right)$ can be calculated in time $O(\mathrm{size}(d)\,\mathrm{size}(n))$, where size means binary length (see, for example, [Cohen 00]), and that both d and n are of size $O(\log X)$ in this case. On summing, we see that the time needed here is therefore

$$\ll \frac{X^2 \log(X)^2 \mathsf{digits}}{\Delta X}.$$

So, by choosing $\Delta X \gg X^{1/2+\epsilon}$, we can make the overall time spent on computing the Kronecker symbol $o(X^{3/2})$. As ΔX increases, there is a tradeoff between spending less time on the character computation and having larger arrays, similar to our computation of gcd's in the $d < 0$ case. There is a definite advantage, depending on the particular hardware, to having smaller arrays, i.e., smaller ΔX, to reduce the number of calls to move data from RAM into cache. On our hardware, and in our range $0 < d < 1.3 \times 10^{10}$, we found that a value of $\Delta X = 10^6$ worked well.

Thus, the bulk of the work is spent on looping, for each block, through n and d, looking up the precomputed character values, computing the normalized incomplete gamma function $G(1/4, \pi n^2/d)$ to given precision, and updating the corresponding value of $L(1/2, \chi_d)$ by the amount $\chi_d(n) f(n)$.

The kind of work and operations required is thus very similar to our approach for the $d < 0$ case, with the handling of characters similar to our handling of gcd's, and the approach to computing the incomplete gamma function similar to that of the K-Bessel function.

However, there is one significant difference in the two methods. For $d < 0$, equation (3–11) tells us that our Epstein zeta function method loops through

$$\frac{\pi}{36\zeta(3)} X^{3/2} \approx 0.0726 X^{3/2}$$

triples a, b, c. Not only is the constant 0.0726 small, but the desired precision does not affect the number of triples required. Precision becomes a factor only in regard to computing the particular contribution from each triple, for example the number of terms needed for the various K-Bessel Taylor series expansions.

But in the present case of the smooth approximate functional equation, both the length of the sum and the amount of work needed to compute the individual terms of the sum depend on the desired precision. Hence the main difference in these two approaches is the length of the sum.

In the case of $d > 0$, the length of the main d, n loops, summed over all blocks of length ΔX, is quantified by

$$L_{\mathrm{pos}} = \sum_{m \le X/\Delta X} \sum_{n \le M} \sum_{(m-1)\Delta X < d \le m\Delta X} 1,$$

with M given by (3–27). Simplifying the two inner sums, we see that this quantity is easily estimated to be asymptotically

$$(\Delta X)^{3/2} \sqrt{\frac{\log(10) \cdot \mathsf{digits}}{\pi}} \sum_{m \le X/\Delta X} \sqrt{m} \sim \frac{2}{3} \sqrt{\frac{\log(10) \cdot \mathsf{digits}}{\pi}} X^{3/2}.$$

So, if $\mathsf{digits} = 16$, then $L_{\mathrm{pos}} \approx 2.28 X^{3/2}$, which is more than twenty times larger than the number of triples, $0.0726 X^{3/2}$, considered for $d < 0$.

It is impossible to precisely pin down, theoretically, the constant-factor savings in the runtime of our method for $d < 0$ compared to the approach used for $d > 0$, since it depends on the speed of the various arithmetic and memory operations on particular hardware. Furthermore, these are not easily quantifiable, since they change according to how the various resources of the machine are being used at a given moment. Another obstacle to a precise comparison is that one would need to take into account implementation choices made by the programmer and also by the compiler at the minutest level.

Nonetheless, a rough comparison between the lengths of the main loops involved, i.e., $2.28X^{3/2}$ for $d > 0$ and $0.0726X^{3/2}$ for $d < 0$ (see (3–11)), does reflect the different runtimes as compared experimentally.

We ran our computation for $d < 0$ on `mod.math.washington.edu`, which is a Sun Fire X4450, dating from 2008, with 24 Intel Xeon X7640 2.66-GHz CPUs (we used 12 of them), and 128 GB RAM. Our computation for $d > 0$ was carried out on `pilatus.uwaterloo.ca`, which is an older SGI Altix 3700 machine, dating from around 2003, with 64 Intel Madison Itanium CPUs (we used 55 of these) running at 1.3 GHz, and 192 GB of RAM.

Our computation, for $d > 0$, took roughly 18.9 CPU years, and about 3.9 CPU years for $d < 0$. Recall that we went up to $0 < -d < 5 \times 10^{10}$, whereas for $d > 0$, we managed to get to 1.3×10^{10}. So not only did our computation for $d < 0$ take much less CPU time, but we went significantly further. To make this more meaningful, we should compare intervals of similar length, i.e., the subset of $0 < -d < 1.3 \times 10^{10}$. This interval required 0.4 CPU years, i.e., about 47 times faster than our computation for the interval $0 < d < 1.3 \times 10^{10}$. However, because different machines were used for $d < 0$ and $d > 0$, we should compensate by dividing the time used for $d > 0$ by a factor of 2.5 to account for the fact that these d were handled on an older and slower machine. The value of 2.5 was determined by rerunning select blocks of $d > 0$ on both machines, using the same C++ code, and comparing their runtimes, which were about 2 to 3 times faster on the newer machine. Therefore, dividing 47 by 2.5, our code ran around 20 times faster for $d < 0$ than it did for $d > 0$, consistent with our rough expectations based on the lengths in both methods of the main loops.

ACKNOWLEDGMENTS

Support for work on this paper was provided by the National Science Foundation under awards DMS-0757627 (FRG grant), an NSERC Discovery Grant, and an OGS Scholarship.

REFERENCES

[Bordellès 07] O. Bordellès. "A Note on the Average Order of the gcd-sum Function." *Journal of Integer Sequences* 10 (2007), Article 07.3.3.

[Broughan 01] K. A. Broughan. "The gcd-Sum Function." *J. Integer Sequences* 4 (2001), Article 01.2.2.

[Chamizo and Iwaniec 98] F. Chamizo and J. Iwaniec. "On the Gauss Mean-Value Formula for Class Number." *Nagoya Math. J.* 151 (1998), 199–208.

[Chowla and Selberg 67] S. Chowla and A. Selberg. "On Epstein's Zeta-Function." *J. Reine Angew. Math.* 227 (1967), 86–110.

[Cohen 00] H. Cohen. *A Course in Computational Algebraic Number Theory*, Graduate Texts in Math. 138. Springer, 2000.

[Conrey and Farmer 00] J. B. Conrey and D. W. Farmer. "Mean Values of L-Functions and Symmetry." *Internat. Math. Res. Notices* 17 (2000), 883–908.

[Conrey et al. 05] J. B. Conrey, D. W. Farmer, J. P. Keating, M. O. Rubinstein, and N. C. Snaith. "Integral moments of ζ and L-Functions." *Proceedings of the London Math. Soc. (3)* 91 (2005), 33–104.

[Davenport 00] H. Davenport. *Multiplicative Number Theory*, 3rd ed., Graduate Texts in Math. 74. Springer 2000.

[Diaconu et al. 03] A. Diaconu, D. Goldfeld, and J. Hoffstein. "Multiple Dirichlet Series and Moments of Zeta and L-Functions." *Compositio Math.* 139 (2003), 297–360.

[Jutila 81] M. Jutila. "On the Mean Values of $L(1/2, \chi)$ for Real Characters." *Analysis* 1 (1981), 149–161.

[Katz and Sarnak 99] N. M. Katz and P. Sarnak. *Random Matrices, Frobenius Eigenvalues, and Monodromy*, AMS Colloquium Publications. American Math Soc., 1999.

[Keating and Snaith 00] J. P. Keating and N. C. Snaith. "Random Matrix Theory and L-Functions at $s = 1/2$." *Comm. Math. Phys.* 214 (2000), 91–110.

[Kuhleitner 02] M. Kuhleitner, "On the Class Number of Binary Quadratic Forms." *Mathematica Pannonica* 13 (2002), 63–78.

[Landau 66] E. Landau. *Elementary Number Theory*. Chelsea, 1966.

[Rubinstein 05] M. Rubinstein, Michael. "Computational Methods and Experiments in Analytic Number Theory." In Recent Perspectives in Random Matrix Theory and Number Theory, pp. 425–506. Cambridge: Cambridge University Press, 2005.

[Soundararajan 00] K. Soundararajan. "Nonvanishing of Quadratic Dirichlet L-Functions at $s = 1/2$." *Ann. of Math. (2)* 152 (2000), 447–488.

[Young 09] M. Young. "The First Moment of Quadratic Dirichlet L-Functions." *Acta. Arith.* 138:1 (2009), 73–99.

[Young 12] M. Young. "The Third Moment of Quadratic Dirichlet L-Functions." Preprint. arXiv: 1203.4457, 2012.

[Zhang 05] Q. Zhang. "On the Cubic Moments of Quadratic Dirichlet L-Functions." *Math. Res. Lett.* 12 (2005), 413–424.

[Zhang 06] Q. Zhang. *Applications of Multiple Dirichlet Series in Mean Values of L-Functions*, Proceedings of Symposia in Pure Math. 75, American Math. Soc., 2006.

Matthew W. Alderson, Pure Mathematics, University of Waterloo, 200 University Ave W., Waterloo, ON, N2L 3G1, Canada (alderson.matthew@gmail.com)

Michael O. Rubinstein, Pure Mathematics, University of Waterloo, 200 University Ave W., Waterloo, ON, N2L 3G1, Canada (michael.o.rubinstein@gmail.com)