Rendiconti Sem. Mat. Univ. Pol. Torino
Vol. 80, 2022 (2022), 55 – 71

Alessandro Languasco

ON COMPUTING $L'/L(1,\chi)$

Abstract. We describe how to efficiently compute $L'/L(1,\chi)$ using the Fast Fourier Transform algorithm. In particular, we will show how to combine the decimation in frequency (DIF) strategy with the reflection formulae of the special functions involved in the computations. We will also mention how to efficiently compute such special functions.

1. Introduction and motivation

Let q be an odd prime. Our interest in computing the values of $L'/L(1,\chi)$, where $L(s,\chi)$ are the Dirichlet L-functions and χ runs over the non-principal Dirichlet characters mod q, started because of Ihara's conjecture on the Euler-Kronecker constants for cyclotomic fields.

To be able to state the problem we need some definition first. Let K be a number field and let $\zeta_K(s)$ be its Dedekind zeta-function. It is a well known fact that $\zeta_K(s)$ has a simple pole at $s=1$; writing the expansion of $\zeta_K(s)$ near $s=1$ as

$$\zeta_K(s) = \frac{c_{-1}}{s-1} + c_0 + \mathscr{O}(s-1),$$

the *Euler-Kronecker constant of* K is defined as

$$\lim_{s\to 1}\Big(\frac{\zeta_K(s)}{c_{-1}} - \frac{1}{s-1}\Big) = \frac{c_0}{c_{-1}}.$$

In the special case in which $K=\mathbb{Q}(\zeta_q)$ is a prime cyclotomic field, where ζ_q is a primitive q-root of unity, we have that the Dedekind zeta-function verifies $\zeta_{\mathbb{Q}(\zeta_q)}(s) = \zeta(s)\prod_{\chi\neq\chi_0} L(s,\chi)$, where $\zeta(s)$ is the Riemann zeta-function, χ runs over the non-principal Dirichlet characters mod q and χ_0 is the principal Dirichlet character mod q. By logarithmic differentiation, we immediately get that the *Euler-Kronecker constant for the prime cyclotomic field* $\mathbb{Q}(\zeta_q)$ is

$$\mathfrak{G}_q := \gamma + \sum_{\chi\neq\chi_0} \frac{L'}{L}(1,\chi), \tag{1}$$

where γ is the Euler-Mascheroni constant. An extensive study about the properties of $\mathfrak{G}_q$ was started by Ihara [13–15] and carried over from many others; here we are mainly interested in computational problems on $\mathfrak{G}_q$ and hence we just recall the paper by Ford-Luca-Moree [9].

Another interesting quantity related to $\mathfrak{G}_q$ is the Euler-Kronecker constant $\mathfrak{G}_q^+$ of $\mathbb{Q}(\zeta_q+\zeta_q^{-1})$, the maximal real subfield of $\mathbb{Q}(\zeta_q)$. According to eq. (10) of Moree [23] it is defined as

$$\mathfrak{G}_q^+ := \gamma + \sum_{\substack{\chi\neq\chi_0\\ \chi \text{ even}}} \frac{L'}{L}(1,\chi).$$

It is interesting to study the negativity of both $\mathfrak{G}_q, \mathfrak{G}_q^+$ because Ihara conjectured they should be both positive. For $\mathfrak{G}_q$ this was disproved by Ford-Luca-Moree [9] by showing $\mathfrak{G}_{964477901} = -0.182374\ldots$. For $\mathfrak{G}_q^+$, Ihara's conjecture is still an open problem.

After the publication of [9], we started to investigate the possibility to obtain other counterexamples to Ihara's conjecture on $\mathfrak{G}_q$ and, at the same time, to extend the statistics on the values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ since Ford-Luca-Moree computed them only for $3 \le q \le 50000$. By distinguishing the contribution of the odd and even Dirichlet characters in (1), we were able to improve on the performances and on the accuracy of the calculations in [9]. The goal of this paper is to give more details on the algorithms we used to obtain our contributions on the topic.

The main tool we will use is the Fast Fourier Transform (FFT) which is a quite fast, but memory demanding, algorithm. It computes a linear combination of complex exponentials whose coefficients are the values of a given finite sequence $\mathscr{A}$ having N elements. Instead of performing such a summation term by term, which would lead to a computational cost of $\mathscr{O}\left(N^2\right)$ products, the FFT procedure implements a *divide et impera* strategy that recursively uses the *decimation in time* or the *decimation in frequency* ideas, see Section 4. This reduces the computational cost to $\mathscr{O}(N\log N)$ products but requires the storage of at least one copy of the whole sequence $\mathscr{A}$. Hence, roughly speaking, we can say that $\mathscr{O}(N)$ memory positions are required to perform such a computation; in practice, more memory space is in fact needed to keep track of the several steps an implementation of the FFT requires. We also recall that it is not an easy task to implement the FFT algorithm; we refer to the original paper of Cooley-Tukey [5] and to Arndt's book [1, Part III] for more details.

The idea of summing according to the parity of the Dirichlet characters, see Section 3, reduced the runtime-memory needed by the FFT-algorithm thus allowing us to obtain new results on the Euler-Kronecker constants and on other related quantities. In fact, to be able to obtain such results in a reasonable amount of time, we also had to efficiently compute the special functions values needed as FTT-inputs, see Section 5. Combining such ideas we recently obtained the following results:

1. we found three new examples of $\mathfrak{G}_q < 0$ and we extended the knowledge of $\mathfrak{G}_q, \mathfrak{G}_q^+$ by computing them for every $3 \le q \le 10^7$; at the same time we found no cases in which $\mathfrak{G}_q^+ < 0$ for $3 \le q \le 10^7$,

see [18] and a paper in collaboration with Righi [20]. The three new primes q such that $\mathfrak{G}_q < 0$ are listed in Table 5.2;

2. in a collaboration con Lamzouri [17], we studied, both theoretically and computationally, the size of $\min_{\chi\neq\chi_0} |L'/L(1,\chi)|$; we were the first to obtain an upper bound for this quantity. Moreover, exploiting a series of computations for $q \leq 10^7$, we obtained some data about the size of its lower bound;
3. we studied Littlewood's [22] estimates on $|L(1,\chi)|$; both theoretically and computationally, see [19] and a paper in collaboration with Trudgian [21].

For detailed descriptions and proofs of these results we refer to the mentioned papers. Here we just focus on showing the main ideas used in computing $L'/L(1,\chi)$ and $L(1,\chi)$. We start by describing Ford-Luca-Moree's method.

2. How to compute $L'/L(1,\chi)$ (Ford-Luca-Moree's method)

If we do not distinguish between Dirichlet characters' parities, we can use eq. (6.1) and (7.4) of Dilcher [7], as in Ford-Luca-Moree, see eq. (3.2) in [9]. In fact eq. (6.1) of [7] gives

$$L'(1,\chi) = -\sum_{a=1}^{q-1} \chi(a)\gamma_1(a,q),$$

where

$$\gamma_1(a,q) = -\frac{1}{q}\Big(\frac{1}{2}(\log q)^2 + \log q\ \psi\big(\frac{a}{q}\big) + \psi_1\big(\frac{a}{q}\big)\Big),$$

for any $q \geq 1$ and $1 \leq a \leq q$, $\psi(x) = \Gamma'/\Gamma(x)$, Γ is Euler's function,

$$\psi_1(x) = -\gamma_1 - \frac{\log x}{x} - \sum_{m=1}^{+\infty}\Big(\frac{\log(x+m)}{x+m} - \frac{\log m}{m}\Big),$$

and

$$\gamma_1 = \lim_{N\to+\infty}\Big(\sum_{j=1}^{N}\frac{\log j}{j} - \frac{(\log N)^2}{2}\Big) = -0.0728158454835\ldots$$

We also remark here that the rate of convergence of the series of $\psi_1(x)$ is, roughly speaking, about $(\log m)/m^2$. Recalling now

$$L(1,\chi) = -\frac{1}{q}\sum_{a=1}^{q-1}\chi(a)\ \psi\big(\frac{a}{q}\big), \tag{2}$$

by the orthogonality of Dirichlet characters and (2), we obtain eq. (3.2) of [9], *i.e.*,

$$L'(1,\chi) = -(\log q)L(1,\chi) + \frac{1}{q}\sum_{a=1}^{q-1}\chi(a)\,T\big(\frac{a}{q}\big),$$

where $T(x) = \gamma_1 + \psi_1(x)$. Summarising, we finally get

$$\sum_{\chi\neq\chi_0}\frac{L'}{L}(1,\chi) = -(q-2)\log q - \sum_{\chi\neq\chi_0}\frac{\sum_{a=1}^{q-1}\chi(a)\,T(a/q)}{\sum_{a=1}^{q-1}\chi(a)\,\psi(a/q)}. \tag{3}$$

In practical applications of the previous formula we encountered the following computational problems:

1. two special functions (T and ψ) have to be computed at rational points in $(0,1)$;
2. T is just defined with a slowly convergent series and ψ is not available in the C-standard programming language;
3. for $x \to 0^+$, $T(x) \sim \log(1/x)/x$ and $\psi(x) \sim -1/x$; so they both become "large" for x close to 0;
4. $q-1$ values of $T(x)$ and $\psi(x)$ are needed;
5. performing the sum over a is computationally "slow", but it can be done efficiently with a Fast Fourier Transform, as we will see in Section 4.

3. How to compute $L'/L(1,\chi)$ in a different way

Our algorithm uses the following formulae that, according to Deninger [6] and Kanemitsu [16], were first proved in 1883 by Berger [2] and in 1929 by Gut [12].

3.1. Primitive odd Dirichlet character case.

Let q be an odd prime, and let $\tau(\chi) := \sum_{a=1}^{q}\chi(a)\,e(a/q)$, $e(x) := \exp(2\pi i x)$, be the Gauß sum associated with χ. The functional equation for $L(s,\chi)$ gives

$$L(s,\chi) = \frac{1}{\pi i}\Big(\frac{2\pi}{q}\Big)^s\Gamma(1-s)\frac{\tau(\chi)}{\sqrt{q}}\cos\Big(\frac{\pi s}{2}\Big)L(1-s,\overline{\chi})$$

and hence

$$\frac{L'(s,\chi)}{L(s,\chi)} = \log\Big(\frac{2\pi}{q}\Big) - \frac{\gamma'(1-s)}{\Gamma(1-s)} - \frac{\pi}{2}\tan\Big(\frac{\pi s}{2}\Big) - \frac{L'(1-s,\overline{\chi})}{L(1-s,\overline{\chi})},$$

which, evaluated at $s=0$, gives

$$\frac{L'(0,\chi)}{L(0,\chi)} = \log\Big(\frac{2\pi}{q}\Big) + \gamma - \frac{L'(1,\overline{\chi})}{L(1,\overline{\chi})}. \tag{4}$$

By the Lerch identity about values of the Hurwitz zeta-function defined as $\zeta(s,x) = \sum_{n=0}^{+\infty}(n+x)^{-s}$ for $\Re(s)>1$, $x\in(0,1)$, and analytically extended to $\mathbb{C}\setminus\{1\}$, namely:

$$\zeta(0,\frac{a}{q}) = \frac{1}{2} - \frac{a}{q}, \quad \zeta'(0,\frac{a}{q}) = \log\Gamma(\frac{a}{q}) - \frac{1}{2}\log(2\pi),$$

the relation

$$L(s,\chi) = q^{-s}\sum_{a=1}^{q-1}\chi(a)\zeta(s,\frac{a}{q}),$$

and the orthogonality of Dirichlet characters, we get

$$\begin{aligned} L'(0,\chi) &= -\log q\sum_{a=1}^{q-1}\chi(a)\Big(\frac{1}{2}-\frac{a}{q}\Big) + \sum_{a=1}^{q-1}\chi(a)\log\Big(\Gamma(\frac{a}{q})\Big) \\ &= \frac{\log q}{q}\sum_{a=1}^{q-1} a\,\chi(a) + \sum_{a=1}^{q-1}\chi(a)\log\Big(\Gamma(\frac{a}{q})\Big) \\ &= -(\log q)L(0,\chi) + \sum_{a=1}^{q-1}\chi(a)\log\Big(\Gamma(\frac{a}{q})\Big). \end{aligned}$$

In the previous formula we also used that

$$L(0,\chi) = -B_{1,\chi} := -\frac{1}{q}\sum_{a=1}^{q-1} a\,\chi(a), \tag{5}$$

where $B_{1,\chi}$ is the first χ-Bernoulli number. We recall that the χ-Bernoulli numbers are defined as the values attained at $x=0$ of the χ-Bernoulli polynomials $B_{k,\chi}(x)$ given by the following series:

$$\frac{te^{tx}}{e^{qt}-1}\sum_{r=0}^{q-1}\chi(r)e^{rt} = \sum_{k\geq 0} B_{k,\chi}(x)\frac{t^k}{k!},$$

see Cohen [4, Section 9.4.1].

Summarising, by (4)-(5), we obtain

$$\sum_{\chi\,\mathrm{odd}}\frac{L'}{L}(1,\chi) = \frac{q-1}{2}\big(\gamma+\log(2\pi)\big) + \sum_{\chi\,\mathrm{odd}}\frac{1}{B_{1,\overline{\chi}}}\sum_{a=1}^{q-1}\overline{\chi}(a)\log\Big(\Gamma(\frac{a}{q})\Big). \tag{6}$$

3.2. Primitive even Dirichlet character case.

Recall that q is an odd prime. Assume now that $\chi \neq \chi_0$ is a primitive even Dirichlet character mod q. We follow Deninger's notation in [6] by calling $R(x) = -\frac{\partial^2}{\partial s^2}\zeta(s,x)|_{s=0} = \log(\Gamma_1(x))$, $x>0$. We will call $\Gamma_1(x)$ the *Ramanujan-Deninger* gamma function. $R(x)$ is the unique solution in $(0,+\infty)$ of the difference equation $R(x+1) = R(x) + (\log x)^2$, with initial condition $R(1) = -\zeta''(0)$, which is convex in some interval $(A,+\infty)$, $A > 0$, see Theorem 2.3 of Deninger [6].

By eq. (3.5)-(3.6) of [6] we have

$$L'(1,\chi) = \big(\gamma+\log(2\pi)\big)L(1,\chi) + \frac{\tau(\chi)}{q}\sum_{a=1}^{q-1}\overline{\chi}(a)\,R\big(\frac{a}{q}\big), \tag{7}$$

where, see eq. (2.3.2) of [6], the R-function can be expressed for every $x>0$ by

$$R(x) := -\zeta''(0) - S(x),$$

$$S(x) := 2\gamma_1 x + (\log x)^2 + \sum_{m=1}^{+\infty}\Big(\big(\log(x+m)\big)^2 - (\log m)^2 - 2x\frac{\log m}{m}\Big). \tag{8}$$

It is worth remarking that $\psi_1(x) = R'(x)/2$. We have $S(1) = 0$ and $R(1) = -\zeta''(0)$. We further remark that the rate of convergence of the series of $S(x)$ is, roughly speaking, about $(\log m)/m^2$. By the orthogonality of the Dirichlet characters, we immediately get

$$\sum_{a=1}^{q-1}\overline{\chi}(a)R(\frac{a}{q}) = -\sum_{a=1}^{q-1}\overline{\chi}(a)\,S(\frac{a}{q}). \tag{9}$$

For $L(1,\chi)$, we use formula (2) of Proposition 10.3.5 of [3]-[4], and the parity of χ to get

$$L(1,\chi) = 2\frac{\tau(\chi)}{q}\sum_{a=1}^{q-1}\overline{\chi}(a)\log\Big(\Gamma\big(\frac{a}{q}\big)\Big), \tag{10}$$

since $W(\chi) = \tau(\chi)/q^{1/2}$ for even Dirichlet characters, see Definition 2.2.25 of [3]-[4].

Summarising, using (7) and (9)-(10), if χ is an even Dirichlet character mod q, we finally get

$$\sum_{\substack{\chi\neq\chi_0\\ \chi\,\text{even}}}\frac{L'}{L}(1,\chi) = \frac{q-3}{2}\big(\gamma+\log(2\pi)\big) - \frac{1}{2}\sum_{\substack{\chi\neq\chi_0\\ \chi\,\text{even}}}\frac{\sum_{a=1}^{q-1}\overline{\chi}(a)\;S(a/q)}{\sum_{a=1}^{q-1}\overline{\chi}(a)\log\big(\Gamma(a/q)\big)}. \tag{11}$$

Now, we compare the situation in using (6) and (11) instead of (3):

1. two special functions (S and $\log\Gamma$) have to be computed at rational points in $(0,1)$; but evaluating $B_{1,\chi}$ doesn't involve any special function;
2. S is just defined with a slowly convergent series; but $\log\Gamma$ is available in the C-standard programming language;
3. for $x\to 0^+$, $S(x)\sim(\log x)^2$ and $\log\Gamma(x)\sim\log(1/x)$; they are much smaller than $T(x),\psi(x)$ as $x\to 0^+$; this is important to have a better control on the accuracy of the FFT, see Section 4.4;
4. $(q-1)/2$ values of $S(x)+S(1-x)$ are needed, see Section 4; $(q-1)$ values of $\log\Gamma$ are needed;
5. performing the sum over a is computationally "slow" but it can be done efficiently with a FFT, as we will see in Section 4.

In Table 5.1 (from [18]) we inserted the comparison between the two methods; the differences in using the FFT-algorithm will be explained in the next section.

Comparison	**Approach with T** (ψ comp. with GSL)	**Approach with T** (ψ precomp. with PARI/GP)	**Approach with S**
Magnitude of the functions for $x\to 0^+$:	$\psi(x)\sim -1/x$ $T(x)\sim\frac{\log(1/x)}{x}$	$\psi(x)\sim -1/x$ $T(x)\sim\frac{\log(1/x)}{x}$	$\log(\Gamma(x))\sim\log(1/x)$ $S(x)\sim(\log x)^2$
Precomputations (T and S with PARI/GP):			
needed space for storing precomputed values ($\langle g\rangle=\mathbb{Z}_q^*$, $a_k:=g^k \bmod q$):	$q-1$ values of $T(a_k/q)$	$2(q-1)$ values of $T(a_k/q)$ and $\psi(a_k/q)$	$(q-1)/2$ values of $S(a_k/q)+S(1-a_k/q)$
number of `write` operations on hard disks:	$q-1$	$2(q-1)$	$(q-1)/2$
number of `sumnum` or `intnum` calls:	$q-1$	$q-1$	$(q-1)/2$
FFT-step (with `fftw`):			
number of `read` operations on hard disks:	$q-1$	$2(q-1)$	$(q-1)/2$
number of FFTs:	2	2	3
length of FFTs:	both $q-1$	both $q-1$	one of length $q-1$; the others of length $(q-1)/2$
total RAM occupation (in number of long double positions; in-place FFTs):	$2q+2$	$2q+2$	$2q$

Table 5.1: Comparison table; GSL stands for the Gnu Scientific Library [11], for PARI/GP, see [26].

From Table 5.1 it is clear that the approach that uses $T(x)$ beats the one which implements $S(x)$ only in the total number of the needed Fast Fourier Transforms*, but in any other aspect the latter is better.

4. The Fast Fourier Transform settings

Focusing on (3), (6) and (11), we remark that, since q is an odd prime, it is enough to get g, a primitive root of q, and χ_1, the Dirichlet character mod q given by $\chi_1(g)=e^{2\pi i/(q-1)}$, to see that the set of the non-principal characters mod q is $\{\chi_1^j:\ j=1,\dots,q-2\}$. Hence, if, for every $k\in\{0,\dots,q-2\}$,

*In fact the FFTs can be independently performed and hence they can be executed in parallel; this eliminates the unique disadvantage in using the S-function method.

we denote $g^k \equiv a_k \in \{1,\dots,q-1\}$, every summation in (3)-(6) and (11) is of the type

$$\sum_{k=0}^{q-2} e\Big(\frac{\sigma jk}{q-1}\Big) f\Big(\frac{a_k}{q}\Big), \tag{12}$$

where $e(x) := \exp(2\pi i x)$, $j \in \{1,\dots,q-2\}$ is fixed, $\sigma = \pm 1$, and f is a suitable function which assumes real values. As a consequence, such quantities are, depending on σ, the Discrete Fourier Transforms, or its inverse transform, of the sequence $\{f(a_k/q)\colon k = 0,\dots,q-2\}$. This approach was first remarked by Rader [24] and it was already used in Ford-Luca-Moree [9] to speed-up the computation of these quantities via the use of FFT-dedicated software libraries.

Now we recall the main idea used in the Cooley-Tukey [5] FFT algorithm: the *decimation in time* strategy. We describe only the special case in which the length of the transform is even and the input sequence is real; in fact Cooley-Tukey analysis holds in the general case too; we refer to [5] and to Arndt's book [1, Part III] for more details.

4.1. FFT: decimation in time (DIT)

Let $\mathscr{A}_k := f(a_k/q) \in \mathbb{R}$, $k = 0,\dots,q-2$, and let $j = 0,\dots,q-2$ be fixed; remark that $j = 0$ corresponds to $\chi_0 \bmod q$. We can write (12) as

$$\big(\mathscr{F}(\mathscr{A}_k)\big)_j := \sum_{k=0}^{q-2} e\Big(\frac{\sigma jk}{q-1}\Big)\mathscr{A}_k.$$

We see now that the first $m = (q-1)/2$ elements (also called "the left part") of the sequence $\mathscr{F}(\mathscr{A}_k)$ can be written using the sequences $\mathscr{A}_{2k}$ and $\mathscr{A}_{2k+1}$; the same also hold for the second m elements (also called "the right part") of $\mathscr{F}(\mathscr{A}_k)$.
Let $j = t + \delta m$, $k = 0,\dots,m-1$, $\delta \in \{0,1\}$. We have

$$\begin{aligned}\big(\mathscr{F}(\mathscr{A}_k)\big)_j &= \sum_{k=0}^{m-1} e\Big(\frac{\sigma(t+\delta m)2k}{q-1}\Big)\mathscr{A}_{2k} \\ &\quad + e\Big(\frac{\sigma(t+\delta m)}{q-1}\Big)\sum_{k=0}^{m-1} e\Big(\frac{\sigma(t+\delta m)2k}{q-1}\Big)\mathscr{A}_{2k+1}.\end{aligned}$$

If $\delta = 0$ ($j = t \in \{0,\dots,m-1\}$) [left part]:

$$\begin{aligned}\big(\mathscr{F}(\mathscr{A}_k)\big)_j &= \sum_{k=0}^{m-1} e\Big(\frac{\sigma j2k}{q-1}\Big)\mathscr{A}_{2k} + e\Big(\frac{\sigma j}{q-1}\Big)\sum_{k=0}^{m-1} e\Big(\frac{\sigma j2k}{q-1}\Big)\mathscr{A}_{2k+1} \\ &= \big(\mathscr{F}(\mathscr{A}_{2k})\big)_j + e\Big(\frac{\sigma j}{q-1}\Big)\big(\mathscr{F}(\mathscr{A}_{2k+1})\big)_j.\end{aligned}$$

If $\delta = 1$ ($j = t+m \in \{m,\dots,q-2\}$) [right part]:

$$(\mathscr{F}(\mathscr{A}_k))_{t+m} = \sum_{k=0}^{m-1} e\Big(\frac{\sigma t 2k}{q-1}\Big)\mathscr{A}_{2k} - e\Big(\frac{\sigma t}{q-1}\Big)\sum_{k=0}^{m-1} e\Big(\frac{\sigma t 2k}{q-1}\Big)\mathscr{A}_{2k+1}$$
$$= (\mathscr{F}(\mathscr{A}_{2k}))_t - e\Big(\frac{\sigma t}{q-1}\Big)(\mathscr{F}(\mathscr{A}_{2k+1}))_t.$$

So both the left and the right parts of the output $\mathscr{F}(\mathscr{A}_k)$ are suitable combinations of $\mathscr{A}_{2k}$ and $\mathscr{A}_{2k+1}$; but such subsums have half a length of the original one! This is the starting point of a recursion algorithm that leads to compute $\mathscr{F}(\mathscr{A}_k)$, a transform of length $q-1$, in $\mathscr{O}(\log q)$ steps and $\mathscr{O}(q\log q)$ products and sums.

We can use the decimation in time strategy, but in our setting we have to distinguish between the even-index and the odd-index subsequences of the output, not of the input. Luckily, there is another possible strategy we can use; we will show it in the next section.

4.2. FFT: decimation in frequency (DIF)

A better fit with our problem is obtained using the *decimation in frequency* strategy: assuming that in (12) one has to distinguish between the parity of j (hence on the parity of the Dirichlet characters), letting $m = (q-1)/2$, for every $j = 0,1,\dots,q-2$, $\mathscr{A}_k := f(a_k/q)$, we have that

$$\sum_{k=0}^{q-2} e\Big(\frac{\sigma jk}{q-1}\Big)\mathscr{A}_k = \sum_{k=0}^{m-1} e\Big(\frac{\sigma jk}{q-1}\Big)\mathscr{A}_k + \sum_{k=0}^{m-1} e\Big(\frac{\sigma j(k+m)}{q-1}\Big)\mathscr{A}_{k+m}$$
$$= \sum_{k=0}^{m-1} e\Big(\frac{\sigma jk}{q-1}\Big)\big(\mathscr{A}_k + (-1)^j\mathscr{A}_{k+m}\big).$$

Let now $j = 2t+\ell$, where $\ell \in \{0,1\}$ and $t \in \mathbb{Z}$. The previous equation becomes

$$\sum_{k=0}^{q-2} e\Big(\frac{\sigma jk}{q-1}\Big)\mathscr{A}_k = \sum_{k=0}^{m-1} e\Big(\frac{\sigma tk}{m}\Big)e\Big(\frac{\sigma \ell k}{q-1}\Big)\big(\mathscr{A}_k + (-1)^\ell\mathscr{A}_{k+m}\big)$$
$$= \begin{cases} \sum_{k=0}^{m-1} e\big(\frac{\sigma tk}{m}\big)b_k & \text{if } \ell = 0 \\ \sum_{k=0}^{m-1} e\big(\frac{\sigma tk}{m}\big)c_k & \text{if } \ell = 1, \end{cases} \tag{13}$$

where $t = 0,\dots,m-1$, $\sigma = \pm 1$,

$$b_k := \mathscr{A}_k + \mathscr{A}_{k+m} \quad \text{and} \quad c_k := e\Big(\frac{\sigma k}{q-1}\Big)\big(\mathscr{A}_k - \mathscr{A}_{k+m}\big).$$

Hence, if we just need the sum over the even, or odd, Dirichlet characters as for $f(x) = S(x)$, $f(x) = \log\Gamma(x)$ or $f(x) = x$, instead of computing

a sum of length $q-1$ we can evaluate a sum of half a length, applied on a suitably modified sequence according to (13). In this case too this is the starting point of a recursion algorithm that leads to compute $\mathscr{F}(\mathscr{A}_k)$, a transform of length $q-1$, in $\mathscr{O}(\log q)$ steps and $\mathscr{O}(q\log q)$ products and sums. Clearly this represents a gain in both the speediness and the memory occupation in running the actual computer program. Moreover, if the values of $\mathscr{A}_k = f(a_k/q)$ have to be precomputed and stored, this also means that the quantity of information we have to save during the precomputation (which might require a consistent amount of time), and to recall for the FFT algorithm, is reduced by a factor of 2.

4.3. The use of the reflection formulae in the decimation in frequency strategy

It is useful to remark that from $\langle g\rangle = \mathbb{Z}_q^*$ it trivially follows that $g^m \equiv q-1 \bmod q$, where $m=(q-1)/2$. Hence, recalling $a_k \equiv g^k \bmod q$, we obtain $a_{k+m} \equiv g^{k+m} \equiv a_k(q-1) \equiv q-a_k \bmod q$ and, as a consequence, we get

$$\mathscr{A}_{k+m} = f\Big(\frac{a_{k+m}}{q}\Big) = f\Big(\frac{q-a_k}{q}\Big) = f\Big(1-\frac{a_k}{q}\Big). \tag{14}$$

So, inserting the *reflection formula* for $S(x)$, see eq. (3.3) of Dilcher [8], into (13)-(14), for every $k=0,\dots,m-1$ and for $f(x)=S(x)$, using (8), the sequence b_k becomes

$$\begin{aligned} &S\Big(\frac{a_k}{q}\Big)+S\Big(\frac{a_{k+m}}{q}\Big) = S\Big(\frac{a_k}{q}\Big)+S\Big(1-\frac{a_k}{q}\Big)\\ &= \log^2\Big(\frac{a_k}{q}\Big) + \sum_{n=1}^{+\infty}\Big(\Big(\log\big(n+\frac{a_k}{q}\big)\Big)^2+\Big(\log\big(n-\frac{a_k}{q}\big)\Big)^2-2(\log n)^2\Big). \end{aligned} \tag{15}$$

Assuming $f(x)=\log\Gamma(x)$, using (14) and the well known *reflection formula* for Euler's Γ given by $\Gamma(x)\Gamma(1-x)=\pi/\sin(\pi x)$, we obtain

$$\begin{aligned} \log\Big(\Gamma\big(\frac{a_k}{q}\big)\Big)+\log\Big(\Gamma\big(\frac{a_{k+m}}{q}\big)\Big) &= \log\Big(\Gamma\big(\frac{a_k}{q}\big)\Big)+\log\Big(\Gamma\big(1-\frac{a_k}{q}\big)\Big)\\ &= \log\pi-\log\Big(\sin\big(\frac{\pi a_k}{q}\big)\Big), \end{aligned}$$

thus further simplifying the final computation by replacing the Γ-function with the sin-function. Analogously

$$\log\Big(\Gamma\big(\frac{a_k}{q}\big)\Big)-\log\Big(\Gamma\big(1-\frac{a_k}{q}\big)\Big) = 2\log\Big(\Gamma\big(\frac{a_k}{q}\big)\Big)+\log\Big(\sin\big(\frac{\pi a_k}{q}\big)\Big)-\log\pi.$$

The case in which $f(x)=x$ is easier; using again $\langle g\rangle=\mathbb{Z}_q^*$, $a_k\equiv g^k \bmod q$ and $g^m\equiv q-1 \bmod q$, we can write that $a_{k+m}\equiv q-a_k \bmod q$; hence

$$a_k - a_{k+m} = a_k-(q-a_k) = 2a_k-q$$

so that in this case we obtain

$$c_k = e\Big(\frac{\sigma k}{q-1}\Big)\Big(\frac{2a_k}{q}-1\Big)$$

for every $k = 0,\dots m-1$, $m = (q-1)/2$, $\sigma = \pm 1$.

4.4. FFT accuracy

We dedicate this paragraph to discuss about the accuracy of the Fast Fourier Transform. According to Schatzman [25, § 3.4, pp. 1159–1160], the root mean square relative error in the FFT is bounded by

$$\Delta = \Delta(N,\varepsilon) := 0.6\varepsilon(\log_2 N)^{1/2}, \tag{16}$$

where ε is the machine epsilon, $\log_2 x$ denotes the base 2 logarithm and N is the length of the sum. According to the IEEE 754-2008 specification, we can set $\varepsilon = 2^{-64}$ for the *long double precision* of the C programming language. So for the largest case we considered, $q = 50040955631$, see Table 5.2, $N = (q-1)/2$, we get that $\Delta < 1.92\cdot 10^{-19}$. To evaluate the euclidean norm of the error we have then to multiply Δ and the euclidean norms of the sequences listed before:

$$x_k := 2\frac{a_k}{q}-1, \quad y_k := \log\Gamma\Big(\frac{a_k}{q}\Big)+\log\Gamma\Big(1-\frac{a_k}{q}\Big)-\log\pi,$$
$$z_k := \log\Gamma\Big(\frac{a_k}{q}\Big)-\log\Gamma\Big(1-\frac{a_k}{q}\Big), \quad w_k := S\Big(\frac{a_k}{q}\Big)-S\Big(1-\frac{a_k}{q}\Big),$$

where $a_k = g^k \bmod q$, $\langle q\rangle = \mathbb{Z}_q^*$, $k = 0,\dots,N-1$. A straightforward computation gives

$$\|x_k\|_2 = \Big(\frac{(q-1)(q-2)}{6q}\Big)^{1/2} = 91324.47246\dots$$

Hence, recalling that $\|\cdot\|_\infty \le \|\cdot\|_2$, for this sequence we can estimate that the maximal error in its FFT-computation is bounded by $1.75\cdot 10^{-14}$ (long double precision case). Unfortunately, no closed formulas for the euclidean norms of the other involved sequences are known but, using $\|\cdot\|_\infty \le \|\cdot\|_2 \le \sqrt{N}\|\cdot\|_\infty$ and the formulae

$$\|y_k\|_\infty = -\log\sin(\pi/q) = 23.49137\dots,$$
$$\|z_k\|_\infty = 2\log\Gamma\Big(\frac{1}{q}\Big)-\log\Big(\frac{\pi}{\sin(\pi/q)}\Big) = 24.63610\dots,$$
$$\|w_k\|_\infty = S\Big(\frac{1}{q}\Big)+S\Big(1-\frac{1}{q}\Big) = 606.93779\dots,$$

that can be obtained using straightforward computations, we have that the errors in their FFT-computations are all $< 1.85\cdot 10^{-11}$.

We also estimated *in practice* the accuracy in the actual computations using the FFTW [10] software library by evaluating at run-time the quantity $\mathscr{E}_j(w_k) := \|\mathscr{F}^{-1}(\mathscr{F}(w_k)) - w_k\|_j$, $j \in \{2,\infty\}$, $\mathscr{F}(\cdot)$ is the Fast Fourier Transform and $\mathscr{F}^{-1}(\cdot)$ is its inverse transform. We focused our attention on w_k since, between the sequences mentioned before, it has the largest norm and hence the worst error estimates. Theoretically we have that $\mathscr{E}_j(w_k) = 0$; moreover, assuming that the root mean square relative error in the FFT is bounded by $\Delta > 0$, it is easy to obtain

$$\mathscr{E}_2(w_k) < \Delta(2+\Delta)\|w_k\|_2 \quad \text{and} \quad \mathscr{E}_\infty(w_k) < \Delta(2+\Delta)\sqrt{N}\|w_k\|_\infty. \tag{17}$$

For $q = 50040955631$, $N = (q-1)/2$ and $\varepsilon = 2^{-64}$ in (16), we get $\Delta(2+\Delta) < 3.83 \cdot 10^{-19}$ and from (17) we obtain

$$\mathscr{E}_2(w_k) < 3.70 \cdot 10^{-11} \quad \text{and} \quad \mathscr{E}_\infty(w_k) < 3.70 \cdot 10^{-11}, \tag{18}$$

where the first estimate suffers from the lack of theoretical information about $\|w_k\|_2$. Moreover, the actual computations using FFTW for this case gave that $\|w_k\|_2 = 1099611.166707\ldots$,

$$\begin{aligned} &\frac{\mathscr{E}_2(w_k)}{\|w_k\|_2} < 6.01 \cdot 10^{-19}, \quad \mathscr{E}_2(w_k) < 4.21 \cdot 10^{-13} \\ &\text{and} \quad \mathscr{E}_\infty(w_k) < 2.23 \cdot 10^{-16} \end{aligned} \tag{19}$$

that are in agreement with (18). It is worth notice that the last two computed estimates in (19) are much better than the corresponding theoretical ones in (18). We finally remark that the computed estimates for the analogous quantities involving x_k, y_k, z_k are smaller than the ones for w_k described before.

Summarising, we can conclude that about ten decimal digits of our final results are correct. Clearly, for smaller values of q more correct decimal digits are in fact available.

5. Improvements in computing the Ramanujan-Deninger gamma function

In a joint work with Righi [20], we improved upon the efficiency in computing the special function S which is strictly related to the Ramanujan-Deninger gamma function since $\log(\Gamma_1(x)) = -S(x) - \zeta''(0)$ for $x > 0$, see Section 3.2. In [20] we first gave an alternative proof of the following theorem which was originally proved by Dilcher [8].

THEOREM 1. *Let $x \in (0,2)$. Then*

$$S(x) = -2\gamma_1(1-x) + 2\sum_{k=2}^{+\infty} \frac{(1-x)^k}{k}\left[\zeta(k)H_{k-1} + \zeta'(k)\right],$$

where $H_k = \sum_{j=1}^{k} 1/j$, $\zeta(\cdot)$ is the Riemann zeta-function and $\zeta'(\cdot)$ is its first derivative.

Moreover, we also proved the following

COROLLARY 1. *Letting $x \in (0,1)\cup(1,2)$, $n \in \mathbb{N}$, $n \geq 1$ be fixed, and*

$$r_S(x,n) = \left\lceil \frac{(n+2)\log 2 + |\log(1-|1-x|)|}{|\log|1-x||} \right\rceil - 1, \tag{20}$$

we have that there exists $\theta = \theta(x) \in (-1/2, 1/2)$ such that

$$S(x) = -2\gamma_1(1-x) + 2\sum_{k=2}^{r_S(x,n)} \frac{(1-x)^k}{k}\left[\zeta(k)H_{k-1} + \zeta'(k)\right] + |\theta|2^{-n}. \tag{21}$$

In the following analysis we will also need the difference formula for $S(x)$, namely

$$S(x+1) = S(x) - (\log x)^2 \quad \text{for every } x > 0. \tag{22}$$

Thanks to (22), the fact that Theorem 1 holds for every $x \in (0,2)$ means that every value of $S(x)$, $x \in (0,1)$, can be computed in two different ways. Moreover it is clear that the best convergence interval for (20) is for $x \in (1/2, 3/2)$ because $r_S(x,n)$ becomes very large as $x \to 0^+$ and as $x \to 2^-$ while it is smaller for x close to 1. Hence from a computational point of view the optimal solution is the following: if $x \in (1/2,1)$ we will directly compute $S(x)$ using (21) while for $x \in (0,1/2)$ we will shift the problem using (22) and then use Theorem 1 in $(1,3/2)$. Such an argument leads to proving the following two corollaries.

COROLLARY 2. *Let $x \in (0,1/2)$. We have that*

$$S(x) = (\log x)^2 + 2\gamma_1 x + 2\sum_{k=2}^{+\infty} \frac{(-x)^k}{k}\left[\zeta(k)H_{k-1} + \zeta'(k)\right].$$

COROLLARY 3. *Let $x \in (0,1/2)$. Letting further $n \in \mathbb{N}$, $n \geq 1$ be fixed and*

$$r'_S(x,n) := r_S(1+x,n) = \left\lceil \frac{(n+2)\log 2 + |\log(1-x)|}{|\log x|} \right\rceil - 1,$$

where $r_S(u,n)$ is defined in Theorem 1, we have that there exists $\eta = \eta(x) \in (-1/2, 1/2)$ such that

$$S(x) = (\log x)^2 + 2\gamma_1 x + 2\sum_{k=2}^{r'_S(x,n)} \frac{(-x)^k}{k}\left[\zeta(k)H_{k-1} + \zeta'(k)\right] + |\eta|2^{-n}. \tag{23}$$

Combining these results we also obtain the following

PROPOSITION 1 (Reflection formulae for S). *Let $x \in (0,1)$, $x \neq 1/2$, $n \in \mathbb{N}$, $n \geq 2$,*

$$r_1(x,n) = \frac{1}{2}\Big(\Big\lceil \frac{(n+2)\log 2 + |\log(1-x)|}{|\log x|} \Big\rceil - 1 \Big)$$

and

$$r_2(x,n) = \frac{1}{2}\Big(\Big\lceil \frac{(n+2)\log 2 + |\log x|}{|\log(1-x)|} \Big\rceil - 1 \Big).$$

There exists $\theta = \theta(x)$ such that for $0 < x < 1/2$ we have

$$S(x) + S(1-x) = (\log x)^2 + 2\sum_{\ell=1}^{r_1} \frac{x^{2\ell}}{\ell}\big[\zeta(2\ell)H_{2\ell-1} + \zeta'(2\ell)\big] + |\theta| 2^{-n},$$

and, for $1/2 < x < 1$, we obtain

$$\begin{aligned} S(x) + S(1-x) \;=\; & (\log(1-x))^2 \\ & + 2\sum_{\ell=1}^{r_2} \frac{(1-x)^{2\ell}}{\ell}\big[\zeta(2\ell)H_{2\ell-1} + \zeta'(2\ell)\big] + |\theta| 2^{-n}. \end{aligned}$$

Similar results can also be obtained for $T(x)$, $\log\Gamma(x)$, and $\psi(x)$.

It is important to remark that in Proposition 1 only the even-index coefficients are present because the others annihilate in summing (21) and (23). This cancellation phenomenon further improves upon the performances in the FFT applications because, after having fixed an accuracy Δ, the number of operations needed to compute $S(a_k/q) + S(1-a_k/q)$ is reduced by a factor of 2 (see the definitions of r_1 and r_2) if compared with the one needed to compute directly $S(a_k/q)$ with (21) or (23). Moreover, a further gain of a factor 2 follows from having halved the length of the sequence to be transformed, see Section 4.2.

Combining the results previously described in this section, we were able to reduce the amount of time to obtain $S(a_k/q) + S(1-a_k/q)$ by a factor of 9000 (using 80 binary digits as data format, i.e., the long double type of the C programming language) with respect to the S-series computation in (15). That allowed us to obtain the results summarized in Table 5.2; we remark that the largest case $q = 50040955631$ was extremely hard to handle since its FFT required ≈ 3.2TB of RAM to be performed. Hence, to overcome this problem we used the hard disk instead, but, clearly, the running time of such a case was heavily affected by this. The boldfaced results are the known cases of negativity for $\mathfrak{G}_q$; the first is by Ford-Luca-Moree [9], the second and the third are in [18] and the last one is in [20].

q	$\mathfrak{G}_q$	$\mathfrak{G}_q^+$	FFT exec time
193894451	0.662110…	9.607705…	4m. 29s.
212634221	1.435141…	11.883540…	4m. 28s.
251160191	1.912681…	11.785574…	2m. 53s.
538906601	1.474911…	12.957235…	11m. 56s.
964477901	**−0.182374…**	10.402224…	23m. 13s.
1139803271	0.768538…	8.313111…	27m. 56s.
1217434451	0.877596…	12.946690…	29m. 16s.
1806830951	0.880396…	11.973128…	47m. 48s.
2488788101	0.424880…	12.248837…	103m. 08s.
2830676081	1.254528…	12.438044…	89m. 59s.
2918643191	0.302793…	12.573983…	87m. 49s.
7079770931	1.544698…	14.301772…	742m. 09s.
9109334831	**−0.248739…**	12.128187…	311m. 28s.
9854964401	**−0.096465…**	12.807752…	326m. 03s.
50040955631	**−0.165953…**	13.897647…	two weeks

Table 5.2: Some results obtained on $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$.

6. Final words

The problem of computing $L'/L(1,\chi)$ requires the use of the Fast Fourier Transform which is a quite fast, but memory demanding, algorithm. We have shown here how to reduce its memory requirements by exploiting the reflection formulae for the special functions involved; moreover, we also have shown how to reduce the computational cost of evaluating such special functions. Combining these results, and using FFTW [10], a software library that performs the Fast Fourier Transform, we were able to greatly extend the available data on the values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ and also to study the size of the values of $L'/L(1,\chi)$ and $L(1,\chi)$, where χ is a non principal Dirichlet character modulo an odd prime $q \leq 10^7$.

Acknowledgements. Some of the calculations here described were performed using the University of Padova Strategic Research Infrastructure Grant 2017: "CAPRI: Calcolo ad Alte Prestazioni per la Ricerca e l'Innovazione", `capri.dei.unipd.it` and on the cluster of the University of Padova, Dipartimento di Matematica "Tullio Levi-Civita", `computing.math.unipd.it/highpc`. The author is grateful for having had such computing facilities at his disposal. I would also like to thank the referee for his/her remarks and suggestions.

References

[1] ARNDT J., *Matters computational. Ideas, algorithms, source code*, Springer, 2011.

[2] BERGER A., *Sur une sommation des quelques séries*, Nova Acta Reg. Soc. Sci. Ups. **12** (1883), 29–31.

[3] COHEN H., *Number Theory. Volume I: Tools and Diophantine Equations*, Springer GTM, vol. 239, 2007.

[4] COHEN H., *Number Theory. Volume II: Analytic and Modern Tools*, Springer GTM, vol. 240, 2007.

[5] COOLEY J.W, TUKEY J.W., *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. **19** (1965), 297–301.

[6] DENINGER C., *On the analogue of the formula of Chowla and Selberg for real quadratic fields*, J. Reine Angew. Math. **351** (1984), 171–191.

[7] DILCHER K., *Generalized Euler constants for arithmetical progressions*, Math. Comp. **59** (1992), 259–282.

[8] DILCHER K., *On generalized gamma functions related to the Laurent coefficients of the Riemann zeta function*, Aequationes Math. **48** (1994), 55–85.

[9] FORD K., LUCA F., MOREE P., *Values of the Euler ϕ-function not divisible by a given odd prime, and the distribution of Euler-Kronecker constants for cyclotomic fields*, Math. Comp. **83** (2014), 1447–1476.

[10] FRIGO M., JOHNSON S.G., *The Design and Implementation of FFTW3*, Proc. IEEE **93** (2), 216–231 (2005). The C library is available at `www.fftw.org`.

[11] GNU SCIENTIFIC LIBRARY, version `2.7`, 2021. Available from `www.gnu.org/software/gsl`.

[12] GUT M., *Die Zetafunktion, die Klassenzahl und die Kronecker'sche Grenzformel eines beliebigen Kreiskorpers*, Comment. Math. Helv. **1** (1929), 160–226.

[13] IHARA Y., *The Euler-Kronecker invariants in various families of global fields*, in "Algebraic Geometry and Number Theory: In Honor of Vladimir Drinfeld's 50th Birthday", V. Ginzburg (ed.), Progress in Mathematics **850**, Birkhäuser, 2006, pp. 407–451.

[14] IHARA Y., *On "M-functions" closely related to the distribution of L'/L-values*, Publ. Res. Inst. Math. Sci. **44** (2008), 893–954.

[15] IHARA Y., MURTY V.K., SHIMURA M., *On the logarithmic derivatives of Dirichlet L-functions at $s = 1$*, Acta Arith. **137** (2009), 253–276.

[16] KANEMITSU S., *On evaluation of certain limits in closed form*, in "Théorie des nombres, Proceedings of the International Number Theory Conference", Université Laval, July 5-18, 1987, J.M. de Koninck and C. Levesque (eds.), De Gruyter, 1989, pp. 459–474.

[17] LAMZOURI Y., LANGUASCO A., *Small values of $|L'/L(1,\chi)|$*, Experiment. Math., electronically publ. on September 3, 2021, DOI:`10.1080/10586458.2021.1927255`, (to appear in print).

[18] LANGUASCO A., *Efficient computation of the Euler-Kronecker constants for prime cyclotomic fields*, Res. Number Theory **7** (2021), Paper No. 2, 22 pp.

[19] LANGUASCO A., *Numerical verification of Littlewood's bounds for $|L(1,\chi)|$*, J. Number Theory **223** (2020), 12–34.

[20] LANGUASCO A., RIGHI L., *A fast algorithm to compute the Ramanujan-Deninger Gamma function and some number-theoretic applications*, Math. Comp. **90** (2021), 2899–2921.

[21] LANGUASCO A., TRUDGIAN T.S., *Uniform effective estimates for $|L(1,\chi)|$*, J. Number Theory **236** (2022), 245–260.

[22] LITTLEWOOD J.E., *On the class number of the corpus $P(\sqrt{-k})$*, Proc. London Math. Soc. **27** (1928), 358–372.

[23] MOREE P., *Irregular Behaviour of Class Numbers and Euler-Kronecker Constants of Cyclotomic Fields: The Log Log Log Devil at Play*, in "Irregularities in the Distribution of Prime Numbers. From the Era of Helmut Maier's Matrix Method and Beyond", J. Pintz and M.Th. Rassias (eds.), Springer, 2018, pp. 143–163.

[24] RADER C.M., *Discrete Fourier transforms when the number of data samples is prime*, Proc. IEEE **56** (1968), 1107–1108.

[25] SCHATZMAN J.C., *Accuracy of the Discrete Fourier Transform and the Fast Fourier Transform*, SIAM J. Sci. Comput. **17** (1996), 1150–1166.

[26] THE PARI GROUP, PARI/GP version 2.13.3, Bordeaux, 2021, `pari.math.u-bordeaux.fr`.

AMS Subject Classification: Primary 33-04, 11-04; secondary 33E20, 11Y16, 11Y60

Alessandro LANGUASCO,
Dipartimento di Matematica "Tullio Levi-Civita", Università di Padova,
Via Trieste 63, 35121 Padova, ITALY.
e-mail: `alessandro.languasco@unipd.it`

Lavoro pervenuto in redazione il 14.03.2022.