Research in Number Theory

RESEARCH

# Efficient computation of the Euler–Kronecker constants of prime cyclotomic fields

Alessandro Languasco*

*Correspondence:
alessandro.languasco@unipd.it
Dipartimento di Matematica, "Tullio Levi-Civita", Università di Padova, Via Trieste 63, 35121 Padova, Italy

**Abstract**

We introduce a new algorithm, which is faster and requires less computing resources than the ones previously known, to compute the Euler–Kronecker constants $\mathfrak{G}_q$ for the prime cyclotomic fields $\mathbb{Q}(\zeta_q)$, where $q$ is an odd prime and $\zeta_q$ is a primitive $q$-root of unity. With such a new algorithm we evaluated $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$, where $\mathfrak{G}_q^+$ is the Euler–Kronecker constant of the maximal real subfield of $\mathbb{Q}(\zeta_q)$, for some very large primes $q$ thus obtaining two new negative values of $\mathfrak{G}_q$: $\mathfrak{G}_{9109334831} = -0.248739\ldots$ and $\mathfrak{G}_{9854964401} = -0.096465\ldots$ We also evaluated $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ for every odd prime $q \le 10^6$, thus enlarging the size of the previously known range for $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$. Our method also reveals that the difference $\mathfrak{G}_q - \mathfrak{G}_q^+$ can be computed in a much simpler way than both its summands, see Sect. 3.4. Moreover, as a by-product, we also computed $M_q = \max_{\chi\neq\chi_0} |L'/L(1,\chi)|$ for every odd prime $q \le 10^6$, where $L(s,\chi)$ are the Dirichlet $L$-functions, $\chi$ run over the non trivial Dirichlet characters mod $q$ and $\chi_0$ is the trivial Dirichlet character mod $q$. As another by-product of our computations, we will provide more data on the generalised Euler constants in arithmetic progressions.

**Keywords:** Euler–Kronecker constants, Generalised Euler constants in arithmetic progressions, Application of the Fast Fourier Transform

**Mathematics Subject Classification:** Primary 11-04, secondary 11Y60

## 1 Introduction

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

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

the *Euler–Kronecker constant of* $K$ is defined as

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

In the special case in which $K = \mathbb{Q}(\zeta_q)$ is a prime cyclotomic field, where $q$ is an odd prime and $\zeta_q$ is a primitive $q$-root of unity, we have that the Dedekind zeta-function satisfies $\zeta_{\mathbb{Q}(\zeta_q)}(s) = \zeta(s)\prod_{\chi\neq\chi_0} L(s,\chi)$, where $\zeta(s)$ is the Riemann zeta-function, $L(s,\chi)$

are the Dirichlet $L$-functions, $\chi$ runs over the non trivial Dirichlet characters mod $q$ and $\chi_0$ is the trivial 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 $\gamma$ is the Euler–Mascheroni constant. Sometimes the quantity $\mathfrak{G}_q$ is denoted as $\gamma_q$ but this conflicts with notations used in literature.

An extensive study of the properties of $\mathfrak{G}_q$ was started by Ihara [15,16] and continued by many others; here we are mainly interested in computational problems involving $\mathfrak{G}_q$ and hence we just recall the paper by Ford et al. [9]. We introduce a new method to compute the Euler–Kronecker constants of prime cyclotomic fields which is faster and uses less computing resources than the ones previously known. The new algorithm requires the values of the *generalised gamma function* $\Gamma_1$ at some rational arguments $a/q \in (0, 1)$. Such a function has, for $q$ large and $a = o(q)$, an order of magnitude exponentially smaller than the ones previously used to determine $\mathfrak{G}_q$, see Sect. 4 below. Moreover, the Fast Fourier Transform (FFT) used in this new approach allows a *decimation in frequency strategy*[1] that leads to gaining a factor $1/2$ in the quantity of needed precomputation operations, in the length of the involved transforms and in their memory usage. Our algorithm uses the formulae in Sect. 3 below that, according to Deninger [6] and Kanemitsu [14], were first proved in 1883 by Berger [1] and in 1929 by Gut [13].

Another interesting quantity related to $\mathfrak{G}_q$ is the Euler–Kronecker constant $\mathfrak{G}_q^+$ for $\mathbb{Q}(\zeta_q + \zeta_q^{-1})$, the maximal real subfield of $\mathbb{Q}(\zeta_q)$. According to Eq. (10) of Moree [20] it is defined as

$$\mathfrak{G}_q^+ := \gamma + \sum_{\substack{\chi \neq \chi_0 \\ \chi \text{ even}}} \frac{L'}{L}(1, \chi). \tag{2}$$

In Sect. 3.4 we will give a formula that leads a to a direct evaluation of $\mathfrak{G}_q^+$ in terms of some special functions values attained at some rationals $a/q \in (0, 1)$. Moreover, in Sect. 3.4 we will use the previously proved relations to see why the quantity $\mathfrak{G}_q - \mathfrak{G}_q^+$ is much easier to compute than both its summands.

During such computations, as a by-product, we also evaluated the related quantity

$$M_q := \max_{\chi \neq \chi_0} \left| \frac{L'}{L}(1, \chi) \right|, \tag{3}$$

see Sect. 5. Other quantities related to $\mathfrak{G}_q$ are the *generalised Euler constants in arithmetic progressions*, sometimes also called *Stieltjes constants in arithmetic progressions*, denoted as $\gamma_k(a, q)$, $k \in \mathbb{N}$, $q \geq 1$, $1 \leq a \leq q$, which are defined by

$$\gamma_k(a, q) := \lim_{N \to +\infty} \left( \sum_{\substack{0 < m \leq N \\ m \equiv a \bmod q}} \frac{(\log m)^k}{m} - \frac{(\log N)^{k+1}}{q(k+1)} \right)$$

[1] We use here this nomenclature since it is standard in the literature on the Fast Fourier Transform, but it could be translated in number theoretic language using suitable properties of Dirichlet characters.

$$= -\frac{1}{q}\left(\frac{(\log q)^{k+1}}{k+1} + \sum_{n=0}^{k}\binom{k}{n}(\log q)^{k-n}\psi_n\Big(\frac{a}{q}\Big)\right), \tag{4}$$

see Eqs. (1.3)–(1.4) and (7.3) of Dilcher [7], where

$$\psi_n(z) := -\gamma_n - \frac{(\log z)^n}{z} - \sum_{m=1}^{+\infty}\left(\frac{(\log(m+z))^n}{m+z} - \frac{(\log m)^n}{m}\right) \tag{5}$$

for $n \in \mathbb{N}$ and $z \in \mathbb{C} \setminus \{0, -1, -2, \dots\}$, $\psi_n(1) = -\gamma_n$, and the *generalised Euler constants* $\gamma_n$ are defined as

$$\begin{aligned}\gamma_n &:= \lim_{N\to+\infty}\left(\sum_{j=1}^{N}\frac{(\log j)^n}{j} - \frac{(\log N)^{n+1}}{n+1}\right)\\ &= \sum_{m=1}^{+\infty}\left(\frac{(\log m)^n}{m} - \frac{(\log(m+1))^{n+1} - (\log m)^{n+1}}{n+1}\right),\end{aligned} \tag{6}$$

by, *e.g.*, Eqs. (3)–(4) of Bohman–Fröberg [3]. Remark that $\gamma_0 = \gamma$. It is worth recalling that the functions $\psi_n(z)$ occur in Ramanujan's second notebook, see [2, Chap. 8, Entry 22].

The quantities in (4) and, as we will see in Sects. 2–3 below, the one in (1), are hence connected with the values of $\psi_n$, $n \geq 1$, which is the logarithmic derivative of $\Gamma_n$, a generalised Gamma function, see Deninger [6], Dilcher [8] and Katayama [18], whose definition for $n = 1$ is given in Sect. 3.2. In some sense we can say that the $\psi_n$-functions, $n \geq 1$, are the analogue of the usual *digamma* function. In the following we will denote as $\psi$ the standard digamma function $\Gamma'/\Gamma$; we also remark that it can be represented as the function $\psi_0$ defined in (5).

The paper is organised as follows. In Sect. 2–3 we will give the derivations of the main formulae we will need in the computations; such proofs are classical and are based on the functional equation for the Dirichlet $L$-functions, see Cohen's books [4,5], for instance. Other useful references for this part are the papers of Deninger [6] and Dilcher [7,8].

In Sect. 4 we will see how to implement the formulae of the previous two sections, starting form a straightforward application of the definitions (1)–(2) of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$; then we will compare the Ford–Luca–Moree approach, based on the formulae of Sect. 2, with our new procedure, based on the formulae of Sect. 3. In particular we will see, in both cases, how to insert the Fast Fourier Transform and we will discuss their precisions, computational costs and memory usages. In Sect. 5 we describe how to compute $M_q$. Section 6 is devoted to provide more data on the generalised Euler constants in arithmetic progressions. Finally, we will provide several tables containing a comparison scheme of the different implementations and the computational results and running times of the different approaches. We will also insert two colored scatter plots for the normalised values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ for every prime $q$, $3 \leq q \leq 10^6$, and two scatter plots about $M_q$ and its normalised values for the same set of primes.

We finally remark that some of the ideas presented here will also be used in a joint work with Pieter Moree, Sumaia Saad Eddin and Alisa Sedunova on the computation of the Kummer ratio of the class number for prime cyclotomic fields, see [19].

## 2 The Ford–Luca–Moree method

Recall that $q$ is an odd prime. If we do not restrict to Dirichlet characters of a prescribed parity, we can use Eqs. (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 $\gamma_1(a,q)$ is defined in (4) which, for $k=1$, becomes

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

for any $q \geq 1$ and $1 \leq a \leq q$, where $\psi$, $\psi_1$ are defined in (5). Again using (5), we define

$$T(x) := \gamma_1 + \psi_1(x) = -\frac{\log x}{x} - \sum_{m=1}^{+\infty}\Big(\frac{\log(x+m)}{x+m} - \frac{\log m}{m}\Big), \tag{7}$$

and, specialising (6), we also have

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

To compute $\gamma_1$ and similar constants with a very large precision, see Sect. 6.3 below. We also remark here that the rate of convergence of the series in (7) is, roughly speaking, about $(\log m)/m^2$. Recalling now Eq. (3.1) of [9], *i.e.*,

$$L(1,\chi) = -\frac{1}{q}\sum_{a=1}^{q-1}\chi(a)\ \psi\big(\frac{a}{q}\big), \tag{8}$$

by the orthogonality of Dirichlet characters and (8), 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 $T(x)$ is defined in (7) (pay attention to the change of sign in (7) with respect to Eq. (3.2) of [9]). Summarising, we finally get

$$\sum_{\chi\neq\chi_0}\frac{L'}{L}(1,\chi) = -(q-2)\log q - \sum_{\chi\neq\chi_0}\frac{\sum_{a=1}^{q-1}\chi(a)\ T(a/q)}{\sum_{a=1}^{q-1}\chi(a)\ \psi(a/q)}. \tag{9}$$

Formula (9) is the one used in the paper by Ford–Luca–Moree [9]. We will now explain how we can compute $\mathfrak{G}_q$ via (1) using the values of the two special functions $\psi$ and $T$, together with the values of the non trivial Dirichlet characters mod $q$.

From a computational point of view it is clear that in (9) we first have to evaluate $T(a/q)$ and $\psi(a/q)$ for every $1 \leq a \leq q-1$. For the $\psi$-values we can rely on the PARI/GP function `psi` or, if less precision is sufficient, we can use the analogous function included

in GSL, the *gnu scientific library* [11][2]. For computing the $T$-values, a task for which there are no pre-defined functions in any software libraries we know, we can use the summing function `sumnum` of PARI/GP; this is the most time-consuming step of the procedure. Using the FFT algorithm to perform the sums over $a$, it is easy to see that computing $\mathfrak{G}_q$ via (9) has a computational cost of $\mathscr{O}(q\log q)$ arithmetical operations together with the cost of computing $q-1$ values of the $\psi$ and $T$ functions. For more details see Sect. 4.

## 3 Another method: distinguishing Dirichlet characters' parities

### 3.1 Primitive odd Dirichlet character case

Recall that $q$ is an odd prime, let $\chi \neq \chi_0$ be a primitive odd Dirichlet character mod $q$ 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 $\chi$. The functional equation for $L(s,\chi)$, see, *e.g.*, the proof of Theorem 3.5 of Gun-Murty-Rath [12], 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

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

which, evaluated at $s=0$, gives

$$\frac{L'}{L}(0,\chi) = \log\Big(\frac{2\pi}{q}\Big) + \gamma - \frac{L'}{L}(1,\overline{\chi}). \tag{10}$$

By the Lerch identity about the values of the Hurwitz zeta-function, see, *e.g.*, Proposition 10.3.5 of Cohen [5], and the orthogonality of Dirichlet characters, we get

$$\begin{aligned} L'(0,\chi) &= -\log q \sum_{a=1}^{q-1}\chi(a)\Big(\frac{1}{2}-\frac{a}{q}\Big) + \sum_{a=1}^{q-1}\chi(a)\log\Big(\Gamma\Big(\frac{a}{q}\Big)\Big) \\ &= \frac{\log q}{q}\sum_{a=1}^{q-1} a\chi(a) + \sum_{a=1}^{q-1}\chi(a)\log\Big(\Gamma\Big(\frac{a}{q}\Big)\Big) \\ &= -(\log q)L(0,\chi) + \sum_{a=1}^{q-1}\chi(a)\log\Big(\Gamma\Big(\frac{a}{q}\Big)\Big), \end{aligned} \tag{11}$$

since, see Proposition 9.5.12 and Corollary 10.3.2 of Cohen [5], we have

$$L(0,\chi) = -B_{1,\chi} := -\frac{1}{q}\sum_{a=1}^{q-1} a\chi(a) \neq 0, \tag{12}$$

where $B_{1,\chi}$ is the first $\chi$-Bernoulli number which is non-zero since $\chi$ is odd.

Summarising, by (10)–(12), we obtain

$$\sum_{\chi \bmod \text{odd}} \frac{L'}{L}(1,\chi) = \frac{q-1}{2}\big(\gamma + \log(2\pi)\big) + \sum_{\chi \bmod \text{odd}} \frac{1}{B_{1,\overline{\chi}}}\sum_{a=1}^{q-1}\overline{\chi}(a)\log\Big(\Gamma\Big(\frac{a}{q}\Big)\Big). \tag{13}$$

[2] GSL provides just a *double precision* (in the sense of the C programming language precision) version of $\psi$; hence this is faster, but less accurate, than the computation of the $\log\Gamma$-values needed in the procedure described in the next section. If we use PARI/GP to precompute and store the $\psi$-values, then the costs of the precomputation and the input/output part of the FFT step have to be doubled, see Table 1.

From a computational point of view, in (13) we need to compute the $\log\Gamma$-values; to do so we can rely on an internal PARI/GP function or, if less precision is sufficient, we can use the analogous function included in the C programming language. We remark that, for $x \to 0^+$, $\log\big(\Gamma(x)\big) \sim \log(1/x)$ and $\psi(x) \sim -1/x$; hence for $q$ large and $a = o(q)$, the values of $\log\big(\Gamma(a/q)\big)$ are exponentially smaller than the ones of $\psi(a/q)$. Moreover, to compute the first $\chi$-Bernoulli number $B_{1,\overline{\chi}}$, defined in (12), we just need an integral sequence.

### 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 writing $R(x) = -\frac{\partial^2}{\partial s^2}\zeta(s,x)|_{s=0} = \log(\Gamma_1(x))$, $x > 0$, where $\zeta(s,x)$ is the Hurwitz zeta function, $s \in \mathbb{C}\setminus\{1\}$. By Eqs. (3.5)–(3.6) of [6] we have

$$L'(1,\chi) = (\gamma + \log(2\pi))L(1,\chi) + \frac{\tau(\chi)}{q}\sum_{a=1}^{q-1}\overline{\chi}(a)\,R\Big(\frac{a}{q}\Big), \tag{14}$$

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

$$R(x) := -\zeta''(0) - S(x), \tag{15}$$

$$S(x) := 2\gamma_1 x + (\log x)^2 + \sum_{m=1}^{+\infty}\Big(\big(\log(x+m)\big)^2 - (\log m)^2 - 2x\frac{\log m}{m}\Big). \tag{16}$$

It is worth recalling that comparing (15)–(16) with (7), we see that $\psi_1(x) = R'(x)/2$ (note the different definition of $\gamma_1$ on p. 174 of Deninger's paper). Using (6) we have $S(1) = 0$ and $R(1) = -\zeta''(0)$. An alternative definition of $S(x)$ for $x > 0$, which will be useful during the computations, is implicitly contained in Eq. (2.12) of Deninger [6]:

$$S(x) = 2\int_0^{+\infty}\Big((x-1)e^{-t} + \frac{e^{-xt} - e^{-t}}{1-e^{-t}}\Big)\frac{(\gamma + \log t)}{t}\,\mathrm{d}t, \quad x > 0. \tag{17}$$

By the orthogonality of the Dirichlet characters, we immediately get

$$\sum_{a=1}^{q-1}\overline{\chi}(a)\,R(a/q) = -\sum_{a=1}^{q-1}\overline{\chi}(a)\,S(a/q). \tag{18}$$

For $L(1,\chi)$, we use formula (2) of Proposition 10.3.5 of Cohen [5] and the parity of $\chi$ to get

$$L(1,\chi) = 2\frac{\tau(\chi)}{q}\sum_{a=1}^{q-1}\overline{\chi}(a)\log\Big(\Gamma\Big(\frac{a}{q}\Big)\Big), \tag{19}$$

since $W(\chi) = \tau(\chi)/q^{1/2}$ for even Dirichlet characters, see Definition 2.2.25 of Cohen [4]. Summarising, using (14) and (18)–(19), if $\chi$ is an even Dirichlet character mod $q$, we finally get

$$\sum_{\substack{\chi\neq\chi_0\\ \chi \text{ even}}}\frac{L'}{L}(1,\chi) = \frac{q-3}{2}\big(\gamma + \log(2\pi)\big) - \frac{1}{2}\sum_{\substack{\chi\neq\chi_0\\ \chi \text{ even}}}\frac{\sum_{a=1}^{q-1}\overline{\chi}(a)\,S(a/q)}{\sum_{a=1}^{q-1}\overline{\chi}(a)\log\big(\Gamma(a/q)\big)}. \tag{20}$$

We remark that in (20) we can reuse the $\log\Gamma$-values already needed in (13). For computing the $S$-values, a task for which there are no pre-defined functions in any software libraries we know, we can use the PARI/GP functions `sumnum` and `intnum`; this is the most time-consuming step of the procedure. We also remark that, for $x \to 0^+$, $S(x) \sim (\log x)^2$ and $T(x) \sim \log(1/x)/x$; hence for $q$ large and $a = o(q)$, the values of $S(a/q)$ are exponentially smaller than the corresponding ones of $T(a/q)$.

Finally we remark that for $S$ and $B_{1,\overline{\chi}}$ we just need the summation over half of the Dirichlet characters involved; hence in both cases in their computation using the Fast Fourier Transform we can implement the so-called *decimation in frequency* strategy that allow us to improve on both the speed and the memory usage of the actual computation, see Sect. 4.1 below. Using the FFT algorithm to perform the sums over $a$, it is easy to see that computing $\mathfrak{G}_q$ via (13) and (20) has a computational cost of $\mathscr{O}(q\log q)$ arithmetical operations together with the cost of computing $q-1$ values of the $\log\Gamma$-function and $(q-1)/2$ decimated in frequency values of the $S$-function[3].

### 3.3 On $\mathfrak{G}_q^+$: the constant attached to the maximal real subfield of $\mathbb{Q}(\zeta_q)$

It is a consequence of the computations in this section that the Euler–Kronecker constant $\mathfrak{G}_q^+$ for $\mathbb{Q}(\zeta_q+\zeta_q^{-1})$, the maximal real subfield of $\mathbb{Q}(\zeta_q)$, is directly connected with the $S$-function since, by (2) and (20), we have

$$\mathfrak{G}_q^+ = \frac{q-1}{2}\gamma + \frac{q-3}{2}\log(2\pi) - \frac{1}{2}\sum_{\substack{\chi\neq\chi_0\\ \chi \text{ even}}} \frac{\sum_{a=1}^{q-1}\overline{\chi}(a)\,S(a/q)}{\sum_{a=1}^{q-1}\overline{\chi}(a)\log\big(\Gamma(a/q)\big)}. \tag{21}$$

Hence in this case the relevant information is encoded in the $S$ and $\log\Gamma$ functions. Clearly $\mathfrak{G}_q^+$ can be obtained during the $\mathfrak{G}_q$-computation since it requires a subset of the data needed for getting $\mathfrak{G}_q$. In Fig. 2 you can find a colored scatter plot of its values for every $q$ prime, $3 \le q \le 10^6$.

Moreover, a direct evaluation of $\mathfrak{G}_q^+$ via (21) allow us to use a *decimation in frequency* strategy in the application of the FFT technique to evaluate the sums over $a$, see Sects. 4.1–4.3.

### 3.4 Regarding $\mathfrak{G}_q - \mathfrak{G}_q^+$

By (1)–(2), (13) and (21) it is trivial to get that

$$\mathfrak{G}_q - \mathfrak{G}_q^+ = \sum_{\chi \text{ odd}} \frac{L'}{L}(1,\chi) = \frac{q-1}{2}\big(\gamma+\log(2\pi)\big) + \sum_{\chi \text{ odd}} \frac{1}{B_{1,\overline{\chi}}}\sum_{a=1}^{q-1}\overline{\chi}(a)\log\Big(\Gamma\big(\frac{a}{q}\big)\Big). \tag{22}$$

This reveals that, from a practical point of view, $\mathfrak{G}_q - \mathfrak{G}_q^+$ is much easier to compute with respect to both $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$: this not just because, as for $\mathfrak{G}_q^+$, it requires a subset of the data needed for $\mathfrak{G}_q$ but also because it involves just one special function, $\log\Gamma$, which is directly available in many software libraries and in the C programming language.

[3] An explanation for this fact can be found towards the end of Sect. 4.1.

In this case too, a direct evaluation of $\mathfrak{G}_q - \mathfrak{G}_q^+$ via (22) allow us to use a *decimation in frequency* strategy in the application of the FFT technique to evaluate the sums over $a$, see Sects. 4.1–4.3. Some computational data about this quantity are also included in [19].

## 4 Comparison of methods, results and running times

First of all we notice that PARI/GP, v. 2.11.4, has the ability to generate the Dirichlet $L$-functions (and many other $L$-functions) and hence the computation of $\mathfrak{G}_q$, $\mathfrak{G}_q^+$ and $M_q$ can be performed using (1)–(3) with few instructions of the gp scripting language. This computation has a linear cost in the number of calls of the `lfun` function of PARI/GP and, at least on our Dell Optiplex desktop machine, it is slower than both the procedures we are about to describe.

Comparing (13) and (20) with (9), we see that in both cases we can rely on pre-defined functions to compute either the $\log\big(\Gamma(a/q)\big)$-values or the $\psi(a/q)$-values, $1 \le a \le q-1$, and finally we have to evaluate the $T$ and $S$ functions respectively involved. We recall that, when taking $q$ very large, it is relevant to know their order of magnitude for $x \to 0^+$; it is easy to verify that $\log\big(\Gamma(x)\big) \sim \log(1/x)$, $S(x) \sim (\log x)^2$, $\psi(x) \sim -1/x$ and $T(x) \sim \log(1/x)/x$. Hence for $x \to 0^+$, we have that $\log\big(\Gamma(x)\big)$ and $S(x)$ are exponentially smaller than $\psi(x)$ and $T(x)$; a fact that will lead to a more accurate result when using a fixed precision in the final step of the computation. Another difference is that, for the odd Dirichlet characters, the first $\chi$-Bernoulli number in Eq. (13) does not involve any special function, but just an integral sequence. So it seems reasonable to compare the following two approaches:

(a) use the $T$-series formulae and the $\psi$-values as in [9]; in this case we have two possible alternatives to evaluate the $\psi$-function: using GSL (gaining in speed but losing in precision) or using PARI/GP (with a much better precision, but doubling the needed hard disk storage and the number of input/output operations on the hard disk);
(b) use the $S$-function formulae for the even Dirichlet characters case and the first $\chi$-Bernoulli number for the odd one; remark that in both cases we have to evaluate a sum of the log $\Gamma$-values.

This way we can extend the computation performed in [9], not only because we are developing a different implementation of the same formulae, but also because we can solve the problem in an alternative way which is faster, needs less computing resources, and uses functions having a much smaller order of magnitude, see Table 1 for a summary of these facts. In the computation we will use the PARI/GP scripting language to exploit its ability to accurately evaluate the series and integrals involved in the definition of the $T$ and $S$ functions, defined respectively in (7) and (16)–(17), via the functions `sumnum` or `intnum`.

### 4.1 Using the FFT algorithm

We also remark that the procedures (a)–(b) trivially require a quadratic number of arithmetical operations to perform the computations in (9), (13) and (20), but this can be improved by using the FFT algorithm and the following argument. Focusing on (9), (13) and (20), we remark that, since $q$ is prime, it is enough to get $g$, a primitive root of $q$, and $\chi_1$, the Dirichlet character mod $q$ given by $\chi_1(g) = e^{2\pi i/(q-1)}$, to see that the set of the non

trivial characters mod $q$ is $\{\chi_1^j : j = 1, \dots, q-2\}$. Hence, if, for every $k \in \{0, \dots, q-2\}$, we denote $g^k \equiv a_k \in \{1, \dots, q-1\}$, every summation in (9)–(13) and (20) is of the type

$$\sum_{k=0}^{q-2} e\Big(\frac{\sigma jk}{q-1}\Big) f\Big(\frac{a_k}{q}\Big), \tag{23}$$

where $e(x) := \exp(2\pi i x)$, $j \in \{1, \dots, q-2\}$, $\sigma = \pm 1$, and $f$ is a suitable function which assumes real values. As a consequence, such quantities are, depending on $\sigma$, the Discrete Fourier Transforms, or its inverse transformations, of the sequence $\{f(a_k/q) : k = 0, \dots, q-2\}$. This idea was first formulated by Rader [22] and it was already used in [9] to speed-up the computation of these quantities via the use of FFT-dedicated software libraries.

For the approach b) we can also use the *decimation in frequency* strategy: assuming that in (23) one has to distinguish between the parity of $j$ (hence on the parity of the Dirichlet characters), letting $m = (q-1)/2$, for every $j = 0, 1, \dots, q-2$ we have that

$$\begin{aligned}\sum_{k=0}^{q-2} e\Big(\frac{\sigma jk}{q-1}\Big) f\Big(\frac{a_k}{q}\Big) &= \sum_{k=0}^{m-1} e\Big(\frac{\sigma jk}{q-1}\Big) f\Big(\frac{a_k}{q}\Big) + \sum_{k=0}^{m-1} e\Big(\frac{\sigma j(k+m)}{q-1}\Big) f\Big(\frac{a_{k+m}}{q}\Big)\\ &= \sum_{k=0}^{m-1} e\Big(\frac{\sigma jk}{q-1}\Big)\Big(f\Big(\frac{a_k}{q}\Big) + (-1)^j f\Big(\frac{a_{k+m}}{q}\Big)\Big).\end{aligned}$$

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

$$\begin{aligned}\sum_{k=0}^{q-2} e\Big(\frac{\sigma jk}{q-1}\Big) f\Big(\frac{a_k}{q}\Big) &= \sum_{k=0}^{m-1} e\Big(\frac{\sigma tk}{m}\Big) e\Big(\frac{\sigma \ell k}{q-1}\Big)\Big(f\Big(\frac{a_k}{q}\Big) + (-1)^\ell f\Big(\frac{a_{k+m}}{q}\Big)\Big)\\ &= \begin{cases} \sum_{k=0}^{m-1} e\big(\frac{\sigma tk}{m}\big) b_k & \text{if } \ell = 0\\ \sum_{k=0}^{m-1} e\big(\frac{\sigma tk}{m}\big) c_k & \text{if } \ell = 1,\end{cases}\end{aligned} \tag{24}$$

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

$$b_k := f\Big(\frac{a_k}{q}\Big) + f\Big(\frac{a_{k+m}}{q}\Big) \quad \text{and} \quad c_k := e\Big(\frac{\sigma k}{q-1}\Big)\Big(f\Big(\frac{a_k}{q}\Big) - f\Big(\frac{a_{k+m}}{q}\Big)\Big).$$

Hence, if we just need the sum over the even, or odd, Dirichlet characters as in the procedure b) for $f(x) = S(x)$ or $f(x) = x$, instead of computing an FFT transform of length $q-1$ we can evaluate an FFT of length $(q-1)/2$, applied on a suitably modified sequence according to (24). Clearly this represents a gain in both the speed and the memory usage in running the actual computer program. Moreover, if the values of $f(a_k/q)$ have to be precomputed and stored, this also means that the quantity of information we have to save during the precomputation (which will be the most time consuming part), and to recall for the FFT algorithm, is reduced by a factor of 2.

In Table 1 we give a summary of the main characteristics of both approaches for computing $\mathfrak{G}_q$; it is clear that the one using $T(x)$ beats the one which implements $S(x)$ only in the total number of the needed FFT transforms[4], but in any other aspect the latter is

[4] In fact the FFT transforms can be independently performed and hence they can be executed in parallel; this eliminates the unique disadvantage in using the $S$-function method.

**Table 1 Comparison of the main characteristics of procedures a) and b) to compute $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$**

| Comparison | Procedure a) ($\psi$ comp. with GSL) | Procedure a) ($\psi$ comp. with PARI/GP) | Procedure b) |
|---|---|---|---|
| Magnitude of the functions for $x \to 0^+$: | $\psi(x) \sim -1/x$ | $\psi(x) \sim -1/x$ | $\log(\Gamma(x)) \sim \log(1/x)$ |
| | $T(x) \sim \frac{\log(1/x)}{x}$ | $T(x) \sim \frac{\log(1/x)}{x}$ | $S(x) \sim (\log x)^2$ |
| 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 usage (in number of long double positions; in-place FFTs): | $2q+2$ | $2q+2$ | $2q^\dagger$ |

† But the computation for $\mathfrak{G}_q^+$ requires only $(3q+5)/2$ long double positions; so, reusing a portion of the RAM after the computation of $\mathfrak{G}_q - \mathfrak{G}_q^+$, in the second part of the program we essentially have a gain of about $(q-1)/2$ long double positions for the RAM usage

better. In particular the procedure b) is much faster in the precomputation part since its cost is $\le 1/2$ than approach a)'s one.

### 4.2 Decimation in frequency for the even Dirichlet characters case

We make explicit the form that the sequence $b_k$ defined in (24), assumes in our cases.

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

$$f\Big(\frac{a_{k+m}}{q}\Big) = f\Big(\frac{q-a_k}{q}\Big) = f\Big(1-\frac{a_k}{q}\Big). \tag{25}$$

So, inserting the *reflection formula* for $S(x)$, see Eq. (3.3) of Dilcher [8][5], into (24)–(25), for every $k = 0, \dots, m-1$ and for $f(x) = S(x)$, using (16), the sequence $b_k$ becomes

$$\begin{aligned} S\Big(\frac{a_k}{q}\Big) + S\Big(\frac{a_{k+m}}{q}\Big) &= S\Big(\frac{a_k}{q}\Big) + S\Big(1-\frac{a_k}{q}\Big) = \\ &= \Big(\log\frac{a_k}{q}\Big)^2 + \sum_{n=1}^{+\infty}\Big(\Big(\log\Big(n+\frac{a_k}{q}\Big)\Big)^2 + \Big(\log\Big(n-\frac{a_k}{q}\Big)\Big)^2 - 2(\log n)^2\Big), \end{aligned} \tag{26}$$

[5] Pay attention to the fact that the Deninger $S(x)$-function defined in (15)–(16) is equal to $-2\log(\Gamma_1(x))$ as defined in Proposition 1 of Dilcher [8].

where $a_k \equiv g^k \bmod q$, while, using (17), we obtain

$$S\Big(\frac{a_k}{q}\Big) + S\Big(1-\frac{a_k}{q}\Big) = 2\int_0^{+\infty}\Big(-e^{-t} + \frac{e^{-\frac{a_k}{q}t} + e^{-(1-\frac{a_k}{q})t} - 2e^{-t}}{1-e^{-t}}\Big)\frac{(\gamma+\log t)}{t}\,\mathrm{d}t$$
$$= 2\int_0^{+\infty}\Big(-3 + e^{-t} + e^{\frac{a_k}{q}t} + e^{(1-\frac{a_k}{q})t}\Big)\frac{(\gamma+\log t)}{t(e^t-1)}\,\mathrm{d}t, \qquad (27)$$

in which we exploited the uniform convergence of the involved integrals. To optimise speed and precision, both equations (26)–(27) will be used during the actual computations; when possible we will exploit the exponential decay $e^{-ct}$, with $c = \min(a_k/q, 1-a_k/q)$, of the integrand function in (27) using the PARI/GP function `intnum`. But when the parameter $c$ will become too small to give reliable results, we will switch to apply the PARI/GP function `sumnum` to Eq. (26); in this case, roughly speaking, the decay order is $(\log n)/n^2$.

Hence, thanks to the previous formulae, the number of calls to the `sumnum` or `intnum` functions required in the precomputation of the $S$-values is reduced by a factor of 2 with respect to the ones needed to precompute the $T$-values.

If we are just interested in the computation of $\mathfrak{G}_q^+$, we can directly use (21) in which we can embed (26)–(27) and the following remark about the needed $\log\Gamma$-values. Assuming $f(x) = \log\Gamma(x)$, using (25) and the well-known *reflection formula* $\Gamma(x)\Gamma(1-x) = \pi/\sin(\pi x)$, we obtain

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

thus further simplifying the final computation by replacing the $\Gamma$-function with the sin-function.

### 4.3 Decimation in frequency for the odd Dirichlet characters case

We make explicit the form that the sequence $c_k$ defined in (24), assumes in our cases.

If we are just interested in the computation of $\mathfrak{G}_q - \mathfrak{G}_q^+$, we can directly use (22); using the *reflection formula* $\Gamma(x)\Gamma(1-x) = \pi/\sin(\pi x)$ and arguing as in the previous paragraph, we obtain

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

for every $k = 0,\dots,m-1$, $m = (q-1)/2$, and hence $c_k$ is modified accordingly. In this case the gain of using the previous formula is that the number of needed evaluations of the $\log\Gamma$-function is reduced by a factor of 2.

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

#### 4.4 Computations trivially summing over *a* (slower but with more digits available)

Unfortunately in `libpari` the FFT-functions work only if $q = 2^\ell + 1$, for some $\ell \in \mathbb{N}$. So we had to trivially perform these summations and hence, in practice, this part is the most time consuming one in both the procedures a) and b) since it has a quadratic cost in $q$. Being aware of such limitations, we used PARI/GP (with the trivial way to compute the sum over $a$) to evaluate $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ with these three approaches for every odd prime $q \le 300$, on a Dell OptiPlex-3050 (Intel i5-7500 processor, 3.40GHz, 16 GB of RAM and running Ubuntu 18.04.2) using a precision of 30 digits, see Table 2; we also inserted there the values of $M_q$, defined in (3), for the same set of primes. Such results largely extend the precision of the data in Table 1 on p. 1472 of [9]. The computation of the values of Table 2 needed 19 s using the $S$-function, 33 s using the $T$-function and 51 s using PARI/GP `lfun` function. We also computed the values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$, with a precision of 30 digits, for $q = 1009, 2003, 3001, 4001, 5003, 6007, 7001, 8009, 9001, 10007, 20011, 30011$, as you can see in Table 3. These numbers were chosen to heuristically evaluate how the computational cost depends on the size of $q$. In this case, in the fifth column of Table 3 we also reported the running time of the direct approach, *i.e.* using (1), the third and fourth columns are respectively the running times of the other two procedures. For these values of $q$ it became clear that the computation time spent in performing the sums over $a$ was the longest one. This means that inserting an FFT-algorithm is fundamental to further improve the performances of both the algorithms a)-b). We discuss this in more detail in the next paragraph.

#### 4.5 Computations summing over *a* via FFT (much faster but with less digits available)

As we saw before, for large $q$ the time spent in summing over $a$ dominates the computational cost. So we implemented the use of FFT for this task. We first used the `gp2c` compiler tool to obtain suitable C programs to perform the precomputations of the needed $T$ and $S$-values with 38 digits and save them to the hard disk[6]. Then we passed such values to the C programs which used the `fftw` [10] software library to perform the FFT step. In such a final stage the performance was thousand times faster than the one for the same stage trivially performed; as an example you can compare the running times for $q = 10007$, 20011, 30011 in Tables 3 and 4. The running times for the approaches a) and b) reveal that the latter is faster, mainly because it requires less input operations to gain the stored precomputed information since the FFT works on a set of data of half the length than in the former case[7].

This way we computed the values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ for $q = 40009, 42611, 50021, 60013, 70001, 80021, 90001, 100003, 305741, 1000003, 4178771, 6766811, 10000019, 28227761, 75743411$ with the long double precision, see Table 4. These computations were performed with the Dell OptiPlex machine mentioned before.

Some of these $q$-values were chosen for their size and others with the help of $\mathscr{B}$, the "greedy sequence of prime offsets", http://oeis.org/A135311, in the following way. We define $\mathscr{B}$ using induction, by $b(1) = 0 \in \mathscr{B}$ and $b(n) \in \mathscr{B}$ if it is the smallest integer exceeding $b(n-1)$ such that for every prime $r$ the set $\{b(i) \bmod r : 1 \le i \le n\}$ has at most

[6] If we do not use the GSL to directly compute $\psi$, we need to insert its precomputation here.
[7] If $\psi$ is precomputed using PARI/GP, then the gain ratio in the stored space and in the number of input/output operations is raised to 3/4.

**Table 2 Values of $\mathfrak{G}_q$, $\mathfrak{G}_q^+$ and $M_q$ for every odd prime up to 300 with a precision of 30 digits; computed with PARI/GP, v. 2.11.4 with trivial summing over $a$. Total computation time: for $\mathfrak{G}_q$, $\mathfrak{G}_q^+$: 18 sec. 852 millisec., for $M_q$: 19 sec., 171 millisec. on the Dell Optiplex machine mentioned before**

| $q$ | $\mathfrak{G}_q$ | $\mathfrak{G}_q^+$ | $M_q$ |
|---|---|---|---|
| 3 | 0.945497280871680703239749994 15 ... | 0.577215664901532860606512090 08 ... | 0.368281615970147842633237904 07 ... |
| 5 | 1.720624212513404761695728788 65 ... | 1.404895141617037748597559079 76 ... | 0.827679476715504887991046989 67 ... |
| 7 | 2.087594074717330132815424719 57 ... | 1.957156454449714752713821861 43 ... | 0.693743252999179022242316373 93 ... |
| 11 | 2.415425904283267830342879635 83 ... | 2.662074098904331749066540724 53 ... | 0.649609999423979953636904530 77 ... |
| 13 | 2.610757737417650196997761088 57 ... | 2.899595724147905095595912030 13 ... | 0.696309862992037155840892183 52 ... |
| 17 | 3.581976044097577659271788129 19 ... | 3.231791648851081676892004706 42 ... | 1.362931768573113264398333958 90 ... |
| 19 | 4.790409415714283325907039364 58 ... | 3.367028102269433604229117383 61 ... | 1.568219364154767753049389422 69 ... |
| 23 | 2.611289176188200925507391649 64 ... | 3.566052741863034855064900056 33 ... | 1.073702414398956669938630225 04 ... |
| 29 | 3.093731705994268723162751798 19 ... | 3.774512722918181558375405055 27 ... | 1.371734385840801903285830307 99 ... |
| 31 | 4.314442925267475097707574410 42 ... | 3.740634171316317651639278622 31 ... | 1.413151419110044370783998083 70 ... |
| 37 | 4.304938189957602017985579264 17 ... | 3.883461032371137391355234933 88 ... | 1.295189581010783569152784018 21 ... |
| 41 | 3.971521627921332160282570400 14 ... | 3.900672433315760395384204602 89 ... | 1.296736091989581733537965683 80 ... |
| 43 | 4.378627505746950494137750623 36 ... | 4.374628485113751101508848743 89 ... | 1.411768822400511734894513891 81 ... |
| 47 | 4.799394258907416134527584299 88 ... | 4.783305923740314927360885149 64 ... | 1.395675654252736022921027176 03 ... |
| 53 | 4.337736858597092318696960823 07 ... | 4.067348140939114224154518817 81 ... | 1.306275729037908151496679752 64 ... |
| 59 | 5.433516345385003980776344381 93 ... | 5.749774950987178689857145112 91 ... | 1.818993836789378439893483669 29 ... |
| 61 | 5.071085190576516195958050981 13 ... | 4.719191604481376012234792327 91 ... | 1.418099808894416270354591909 83 ... |
| 67 | 5.292139306628962608734284618 31 ... | 5.494785744092310878944509142 85 ... | 1.670191933031543699217826076 34 ... |
| 71 | 5.255258192818946167720131286 37 ... | 5.024592214370138236034534574 63 ... | 1.474555111002367710110158967 67 ... |
| 73 | 4.066949090447495292016488156 25 ... | 5.566380189044206077731448765 27 ... | 1.782489707995986734472825178 91 ... |
| 79 | 4.998276318170680107894313929 45 ... | 4.313928169838421532348144429 52 ... | 1.346168370278134689185886106 88 ... |
| 83 | 3.033136113436074187164038191 05 ... | 4.061198906480154869549604783 74 ... | 1.345277862379107895018758680 23 ... |
| 89 | 4.164090798889832768808411103 72 ... | 5.448348515554347192619029532 43 ... | 1.616546492741263001567820886 73 ... |
| 97 | 4.891240740403896668307514688 57 ... | 4.445634112563467381863804526 64 ... | 1.602861185700764584803622187 99 ... |
| 101 | 5.297012891509669718878600327 39 ... | 5.933645573877269983057899891 64 ... | 1.518719798570796189123672833 35 ... |
| 103 | 5.144339551252088221133305032 20 ... | 5.533125086309998981540064493 39 ... | 1.560727641654860113439219658 20 ... |
| 107 | 5.458274209970245034216802454 53 ... | 5.357446919595968393326035906 20 ... | 1.555294180869365049785520665 30 ... |
| 109 | 6.906638146264236532194698377 04 ... | 6.286393120608420265872823184 84 ... | 1.653578288279083265828411366 43 ... |
| 113 | 4.021730382578030675783180066 17 ... | 4.713080525530713553444516097 38 ... | 1.514869828893521644270604928 78 ... |
| 127 | 5.088599124153334494232156362 40 ... | 5.284275266416422911087148958 25 ... | 1.555901430405964431937929418 54 ... |
| 131 | 2.836826341588379098602857973 21 ... | 4.291824221623893656690362300 41 ... | 1.437978822925316020895642388 79 ... |
| 137 | 4.937000226143684686919629997 11 ... | 5.172819664013681269522670046 84 ... | 1.539298709048677072574695386 80 ... |
| 139 | 5.889168633998671867263837303 69 ... | 5.156734672677856934562006404 45 ... | 1.588288754789132189152408256 92 ... |
| 149 | 5.983424777695159814502427857 39 ... | 6.357442731454876166821519785 17 ... | 1.559334233877546891709270074 57 ... |
| 151 | 5.042016113528721799145194610 22 ... | 5.667322694103882184417686443 82 ... | 1.481710782448887956422260122 30 ... |
| 157 | 7.408022065722227293508452013 90 ... | 5.677664591009700787520769429 90 ... | 1.529150911596116051591498796 96 ... |
| 163 | 5.929664822887206787554999138 44 ... | 5.542896118725225416698601679 04 ... | 2.168327129283523803864003246 42 ... |
| 167 | 8.033001752688724704675833578 02 ... | 6.803947989582599071088391107 55 ... | 1.566072366567503440302935111 54 ... |
| 173 | 3.384347536532061903442977988 97 ... | 4.743136808666541433188644672 69 ... | 1.542424018287161316447239958 19 ... |
| 179 | 3.862361325499030081121261302 82 ... | 5.590747641966937198103045503 44 ... | 1.600850645940720092933009147 35 ... |
| 181 | 5.141118487768481358101366642 57 ... | 5.524011132387354606988935254 057 ... | 1.656565670950100100410937929 77 ... |
| 191 | 4.692869902014226640035524348 12 ... | 6.216216336830787546878895608 01 ... | 1.694008063354780359921951233 69 ... |
| 193 | 5.163422196739154833200782627 20 ... | 6.335168809703022262487492319 89 ... | 1.721068391514300002180162209 49 ... |
| 197 | 7.551487158966406478864851293 72 ... | 6.724312805477589309119316148 98 ... | 1.584252247048569135919063182 69 ... |
| 199 | 6.473665136093207386994974597 78 ... | 4.978673140268340591188073474 77 ... | 1.520555120301924310371079837 92 ... |
| 211 | 7.736135784245861625328105875 85 ... | 5.439287670777068650275927278 91 ... | 1.588876897235216874773423549 47 ... |
| 223 | 7.817779717859913674713367348 51 ... | 6.976407182678804197903011450 60 ... | 1.578094397879642736893107969 56 ... |
| 227 | 8.080531569512962186970711937 57 ... | 6.164781058335358000888390523 12 ... | 1.614404762782895140900732567 62 ... |
| 229 | 7.162986320580995467457781150 58 ... | 5.193681828252284590625827163 49 ... | 1.643916272227055298540731120 16 ... |
| 233 | 3.119483544851275413031152952 58 ... | 5.482686940351806537613263911 37 ... | 1.565348088656696958635933076 80 ... |
| 239 | 3.999110172078332495126322979 19 ... | 4.898260382205097310911882003 57 ... | 1.835932378953422421377996718 38 ... |
| 241 | 6.037525214010342150657092509 35 ... | 6.910995703490281812622494886 55 ... | 1.744835023093562313286852905 92 ... |
| 251 | 5.043137085023473510428111190 22 ... | 5.855224753672624299063775358 83 ... | 1.606342333563945957614343105 31 ... |
| 257 | 8.169913912327413916702251555 27 ... | 7.414131264917794829415719866 52 ... | 1.529863633953225175713217944 33 ... |
| 263 | 7.303436247368154354143480774 06 ... | 6.887618910781859934526394374 20 ... | 1.618736899100657125610080392 62 ... |
| 269 | 6.260348316665771027352527557 12 ... | 6.335724667412823468768398332 27 ... | 1.586623535830789760129533486 99 ... |
| 271 | 5.977178048548033042237739059 76 ... | 4.916073753783495931270847331 5 ... | 1.511451180460000756473402799 32 ... |
| 277 | 4.592808177140778951647770816 61 ... | 6.073063302395309233144135962 79 ... | 1.729741556752771254274515830 60 ... |
| 281 | 4.664964323662114575052208526 23 ... | 4.990437405425582296122528014 06 ... | 1.605363660707047179182423576 61 ... |
| 283 | 7.150285797410682514092252311 88 ... | 7.049692302705228883474597920 33 ... | 1.556091862961423732333165146 03 ... |
| 293 | 3.384381521219539786584682592 38 ... | 5.384381521219539786584682592 38 ... | 1.585153172442840645283567800 36 ... |

**Table 3 A few other values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ with a precision of 30 digits; computed with PARI/GP, v. 2.11.4 with trivial summing over $a$ [m = minutes, s = seconds]. Computation performed on the Dell Optiplex machine mentioned before**

| $q$ | $\mathfrak{G}_q$ | $\mathfrak{G}_q^+$ | time $T$ | time $S$ | time direct |
|---|---|---|---|---|---|
| 1009 | 8.442135151849299275860694672 7... | 6.273354084432210317218625011 1... | 5s. | 3s. | 14s. |
| 2003 | 5.793421369079363328038498216 2... | 6.993525861141397874661684214 2... | 10s. | 7s. | 39s. |
| 3001 | 8.647465136968386938802345350 9... | 8.645970067298413899893497657 7... | 17s. | 11s. | 1m. 11s. |
| 4001 | 7.003435546203143994356851768 4... | 8.780538009423073587286799384 9... | 24s. | 17s. | 1m. 49s. |
| 5003 | 5.549293004581614227736879540 4... | 7.244022474279106263441233061 7... | 32s. | 23s. | 2m. 36s. |
| 6007 | 8.311610121998483816562903440 3... | 9.874266647242576948689612342 0... | 41s. | 30s. | 3m. 22s. |
| 7001 | 8.505277876100877139316878038 4... | 9.683332773491078644708488054 4... | 52s. | 38s. | 4m. 07s. |
| 8009 | 11.686846391549357535345086996 0... | 11.443142155624708487608710920 6... | 1m. 03s. | 47s. | 5m. 00s. |
| 9001 | 10.109478431838340935822503580 2... | 9.486838883145496276749276000 6... | 1m. 15s. | 57s. | 5m. 56s. |
| 10007 | 12.664612004560692327538935678 3... | 11.060162475902474193330828306 3... | 1m. 27s. | 1m. 07s. | 7m. 12s. |
| 20011 | 10.799680311299920518643040289 9... | 10.548980769217096945967222622 1... | 4m. 30s. | 3m. 43s. | 20m. 01s. |
| 30011 | 10.333079972124024225513606225 5... | 11.012703950054089327849887767 4... | 9m. 19s. | 8m. 11s. | 37m. 28s. |

$r-1$ elements. An equivalent statement, assuming that the prime $k$-tuples conjecture holds, is that $b(n)$ is minimal such that $b(1)=0$ and there are infinitely many primes $q$ with $b(i)q+1$ prime for $2\le i\le n$, $n\ge 2$. Let now

$$m(\mathscr{A}) := \sum_{i=1}^{s} \frac{1}{a_i},$$

where $\mathscr{A}$ is an admissible set, *i.e.*, $\mathscr{A}=\{a_1,\dots,a_s\}$, $a_i\in\mathbb{N}$, $a_i\ge 1$, such that does not exist a prime $p$ such that $p \mid n\prod_{i=1}^{s}(a_i n+1)$ for every $n\ge 1$. Thanks to Theorem 2 of Moree [20], if the prime $k$-tuples conjecture holds and if $\mathscr{A}$ is an admissible set, then $\mathfrak{G}_q < (2-m(\mathscr{A})+o(1))\log q$ for $\gg x/(\log x)^{-|\mathscr{A}|-1}$ primes $q\le x$. Moreover, by Theorem 6 of Moree [20], assuming both the Elliott-Halberstam and the prime $k$-tuples conjectures, if $\mathscr{A}$ is an admissible set then $\mathfrak{G}_q = (1-m(\mathscr{A})+o(1))\log q$ for $\gg x/(\log x)^{-|\mathscr{A}|-1}$ primes $q\le x$.

The greedy sequence of prime offsets $\mathscr{B}$ has the property that any finite subsequence is an admissible set. With a PARI/GP script we computed the first 2089 elements of $\mathscr{B}$ since for $\mathscr{C} := \{b(2),\dots,b(2089)\}$ we get $m(\mathscr{C})>2$. So, if we are looking for negative values of $\mathfrak{G}_q$, it seems to be a good criterion to evaluate $\mathfrak{G}_q$ for a prime number $q$ such that $bq+1$ is prime for many elements $b\in\mathscr{C}$ (clearly it is better to start with the smaller available $b$'s). To be able to measure this fact, we define

$$\nu(q) := \sum_{\substack{2\le i\le 2089;\ b(i)\in\mathscr{C}\\ b(i)q+1 \text{ is prime}}} \frac{1}{b(i)}. \tag{28}$$

Some of the $q$-values written before in this paragraph are such that $\nu(q)>1.15$ so that, thanks to Moree's results already cited, they are good candidates to have a negative Euler–Kronecker constant. The complete list of $q\le 10^{10}$ such that $\nu(q)>1.2$ is towards the end of the PARI/GP script `testseq` that can be downloaded here: http://www.math.unipd.it/~languasc/EK-comput.html.

#### 4.5.1 Data for the scatter plots

After having evaluated the running times of the previous examples, we decided to provide the colored scatter plots, see Figs. 1 and 2, of the normalised values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ (both in long double precision) for every odd prime $q\le 10^6$ thus enlarging the known range of the

**Table 4 A few other values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$; computed with PARI/GP, v. 2.11.4 and `fftw`, v. 3.3.8, with long double precision. The sum over $a$ was performed using the FFT algorithm on the Dell Optiplex machine mentioned before [s = seconds, ms = milliseconds; precomputations of decimated in frequency $S$-values performed on the Optiplex; their computation time is excluded from this table]. *: on the Intel Xeon machine due to a runtime memory error on the Dell Optiplex**

| $q$ | $\mathfrak{G}_q$ | $\mathfrak{G}_q^+$ | time |
|---|---|---|---|
| 10007 | 12.664612 . . . | 11.060162 . . . | 10ms. |
| 20011 | 10.799680 . . . | 10.548981 . . . | 23ms. |
| 30011 | 10.333080 . . . | 11.012704 . . . | 15ms. |
| 40009 | 13.146885 . . . | 13.469520 . . . | 25ms. |
| 42611 | 2.499688 . . . | 8.367404 . . . | 41ms. |
| 50021 | 9.910507 . . . | 11.063741 . . . | 98ms.* |
| 60013 | 12.810360 . . . | 12.671109 . . . | 36ms. |
| 70001 | 12.572765 . . . | 13.428551 . . . | 25ms. |
| 80021 | 14.185633 . . . | 11.617216 . . . | 100ms.* |
| 90001 | 11.819424 . . . | 9.601757 . . . | 33ms. |
| 100003 | 15.166074 . . . | 14.765926 . . . | 69ms. |
| 305741 | 1.650523 . . . | 8.839799 . . . | 198ms. |
| 1000003 | 17.379970 . . . | 15.298449 . . . | 876ms. |
| 4178771 | 0.922855 . . . | 8.909168 . . . | 2s. 613ms. |
| 6766811 | 1.604045 . . . | 10.961044 . . . | 4s. 584ms. |
| 10000019 | 17.087945 . . . | 15.974742 . . . | 6s. 361ms. |
| 28227761 | 2.361562 . . . | 10.153369 . . . | 17s. 996ms. |
| 75743411 | 2.469939 . . . | 12.234097 . . . | 2m. 24s. 217ms. |

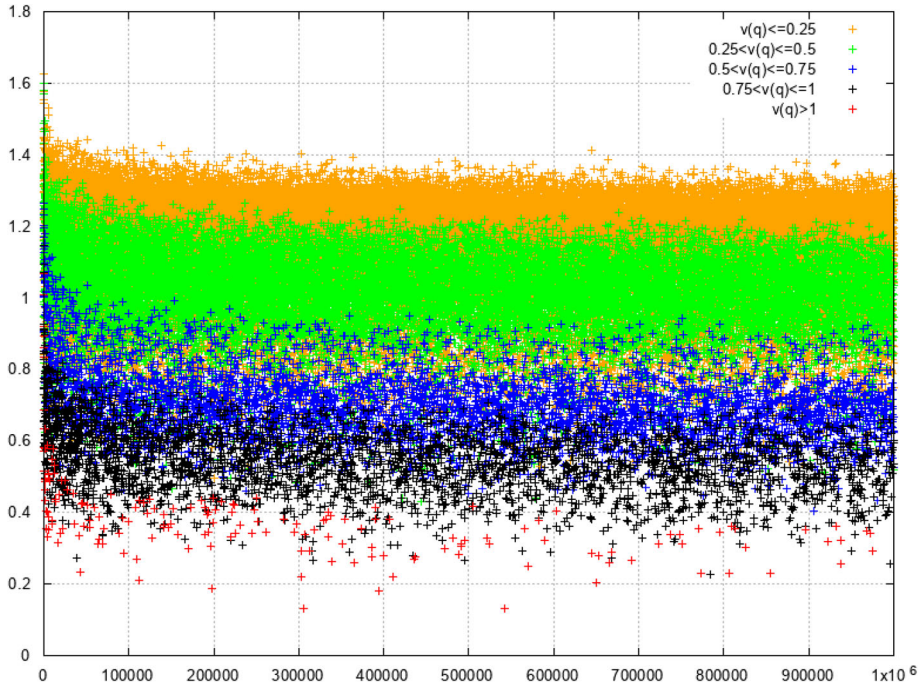


**Fig. 1** The values of $\mathfrak{G}_q/\log q$, $q$ prime, $3 \le q \le 10^6$, plotted using GNUPLOT, v.5.2, patchlevel 8. The minimal value is 0.13067 . . . and it is attained at $q = 305741$; the maximal value is 1.62693 . . . and it is attained at $q = 19$. Orange points satisfy $v(q) \le 0.25$; green points satisfy $0.25 < v(q) \le 0.5$; blue points satisfy $0.5 < v(q) \le 0.75$; black points satisfy $0.75 < v(q) \le 1$; red points satisfy $v(q) > 1$; $v(q)$ is defined in (28)

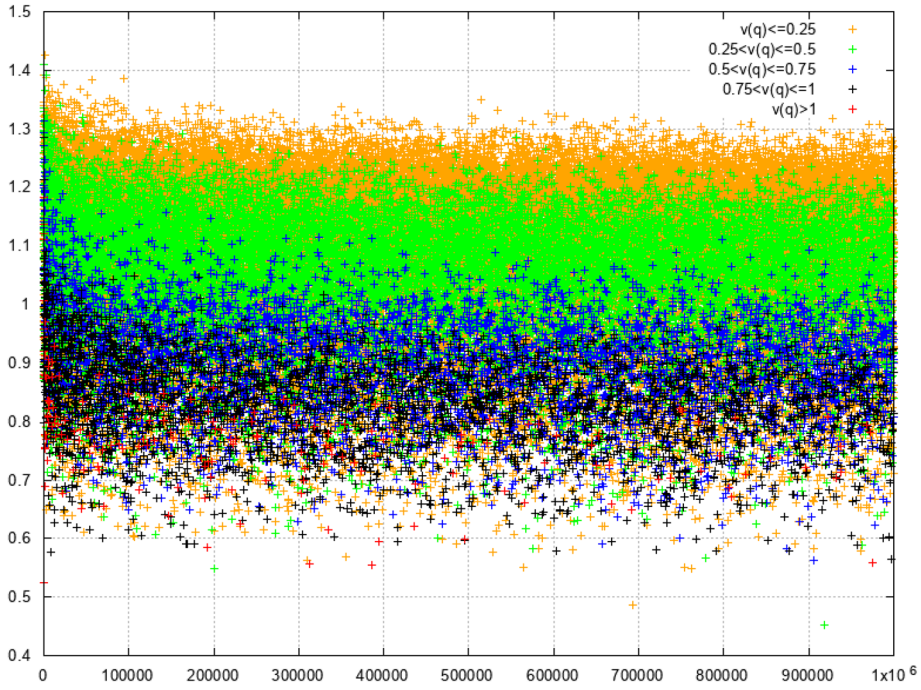


**Fig. 2** The values of $\mathfrak{G}_q^+ / \log q$, $q$ prime, $3 \le q \le 10^6$, plotted using GNUPLOT, v.5.2, patchlevel 8. The minimal value is $0.451468\ldots$ and it is attained at $q = 918787$; the maximal value is $1.42626\ldots$ and it is attained at $q = 2053$. Orange points satisfy $v(q) \le 0.25$; green points satisfy $0.25 < v(q) \le 0.5$; blue points satisfy $0.5 < v(q) \le 0.75$; black points satisfy $0.75 < v(q) \le 1$; red points satisfy $v(q) > 1$; $v(q)$ is defined in (28)

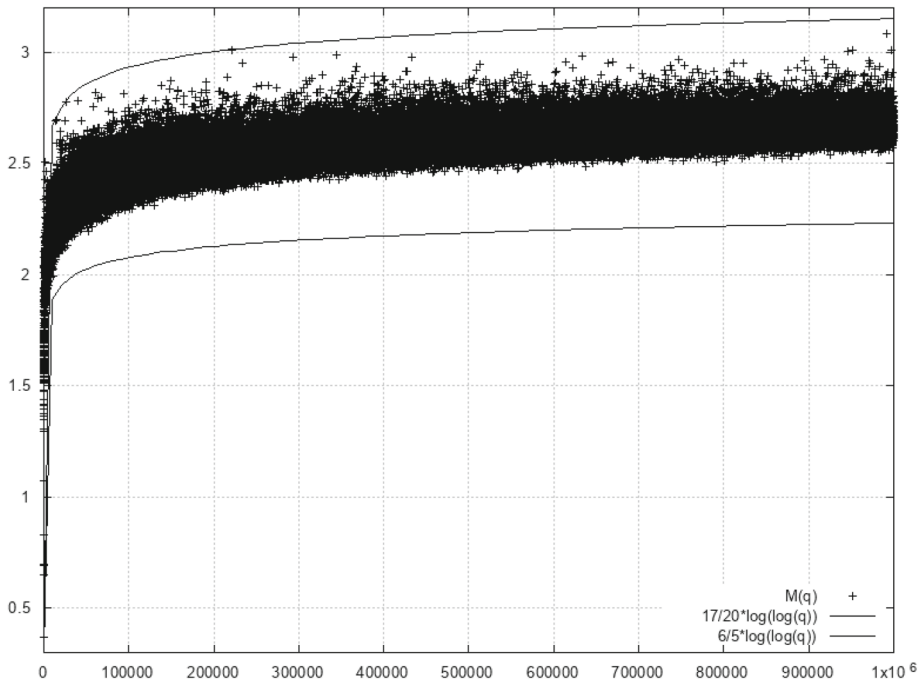


**Fig. 3** The values of $M_q$, $q$ prime, $3 \le q \le 10^6$, plotted using GNUPLOT, v.5.2, patchlevel 8. The minimal value is $0.3682816\ldots$ and it is attained at $q = 3$; the maximal value is $3.085536\ldots$ and it is attained at $q = 991027$. The lines represent the functions $c \cdot \log\log q$, with $c = 17/20$, respectively $c = 6/5$

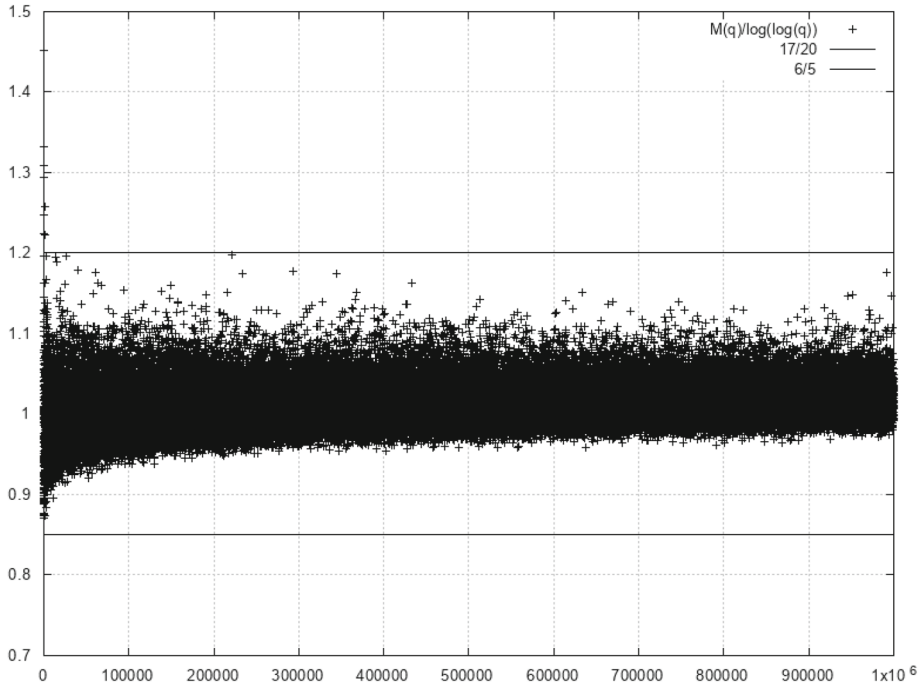


**Fig. 4** The values of $M'_q := M_q/\log\log q$, $q$ prime, $3 \le q \le 10^6$, plotted using GNUPLOT, v.5.2, patchlevel 8. The minimal value is 0.7392305 ... and it is attained at $q = 13$; the maximal value is 3.9158971 ... and it is attained at $q = 3$ (not represented in the plot). $M'_q > 17/20$ for every $13 < q \le 10^6$; $M'_q < 6/5$ for every $1531 < q \le 10^6$. The lines represent the constant functions $c = 17/20$ and $c = 6/5$

**Table 5 A few other values of $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$; computed with PARI/GP, v. 2.11.4 and `fftw`, v. 3.3.8, with long double precision. Boldfaced results are the ones corresponding to known instances of $\mathfrak{G}_q < 0$. The sum over $a$ was performed using the FFT algorithm on the Intel Xeon machine or, for $q = 251160191, 212634221, 1139803271, 7079770931, 9109334831, 9854964401$ on the CAPRI infrastructure mentioned before. [m = minutes, s = seconds; precomputations of decimated in frequency $S$-values performed on the cluster; their computation time is excluded from this table]**

| $q$ | $\mathfrak{G}_q$ | $\mathfrak{G}_q^+$ | 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.** |

data on $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$, see [9]. For performing the needed precomputations of the $S$-values, we used the cluster of the Department of Mathematics of the University of Padova; the cluster setting is described here: http://computing.math.unipd.it/highpc. The minimal value of $\mathfrak{G}_q/\log q$, $3 \le q \le 10^6$, $q$ prime, is $0.13067\ldots$ and it is attained at $q = 305741$, as expected; the maximal value is $1.62693\ldots$ and it is attained at $q = 19$. The minimal value of $\mathfrak{G}_q^+/\log q$, $3 \le q \le 10^6$, $q$ prime, is $0.451468\ldots$ and it is attained at $q = 918787$; the maximal value is $1.42626\ldots$ and it is attained at $q = 2053$. The points $(q, \mathfrak{G}_q/\log q)$ and $(q, \mathfrak{G}_q^+/\log q)$ in Figs. 1 and 2 are colored in orange if $\nu(q) \le 0.25$ (65.65% of the cases), in green if $0.25 < \nu(q) \le 0.5$ (23.62%), in blue if $0.5 < \nu(q) \le 0.75$ (6.29%), in black if $0.75 < \nu(q) \le 1$ (4.21%), and in red if $\nu(q) > 1$ (0.23%). The behaviour of $\mathfrak{G}_q$ is the expected one since the red strip essentially corresponds with its minimal values, while the minima of $\mathfrak{G}_q^+$ seem to be less related to $\nu(q)$; we plan to investigate this phenomenon in the next future. The complete list of numerical results for $\mathfrak{G}_q$ and $\mathfrak{G}_q^+$ can be downloaded at the following web address: https://www.math.unipd.it/~languasc/EKcomput/results.

#### *4.5.2 Computations for larger q*

For values of $q$ larger than 30 millions the precomputation of $T$ and $S$, if performed on a single desktop computer, would require too much time; hence we parallelised them on the cluster previously mentioned. To check the correctness of such computations it is possible to use the following formulae; recalling that $\gamma = 0.577215664901\ldots$ and $\zeta''(0) = -2.006356455908\ldots$, we have that

$$\sum_{a=1}^{q-1} S\Big(\frac{a}{q}\Big) = -\zeta''(0)(q-1) - \log q \log(2\pi) - \frac{(\log q)^2}{2}, \tag{29}$$

$$\sum_{a=1}^{q-1} T\Big(\frac{a}{q}\Big) = \frac{q}{2}(\log q)^2 + \gamma q \log q. \tag{30}$$

Formula (29) is an immediate consequence of Theorem 2.5 of Deninger [6] and formula (30) follows from equation (7.10) of Dilcher [7].

Moreover, for being able to handle very large cases, we used a dedicated `fftw` interface[8] which is able to perform transforms whose length is greater than $2^{31} - 1$.

In this way we were able to obtain an independent confirmation of Theorem 4 of [9] getting $\mathfrak{G}_{964477901} = -0.18237472563711916085\ldots$, since we computed it using the quadruple precision. At the same time we also got $\mathfrak{G}^+_{964477901} = 10.402223382428263536 94\ldots$ To do so we first split the computation, with a precision of 38 digits, of the needed decimated in frequency values of $S$ in 49 subintervals $I_j$ of size $10^7$ each (for $T$ we would need 97 intervals of such a length); the computation time required for each $I_j$ was on average about 1600 min on one of the cluster's machines. Then we passed such values to the programs that performed the FFT-step and got the final results. This last part needed about 23 min (long double precision) or 522 min (quadruple precision) of computation time on an Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, with 160 GB of RAM, running Ubuntu 16.04. A similar procedure let us to get analogous computation times for the long double precision evaluation of $\mathfrak{G}_{1217434451} = 0.877596\ldots$ and $\mathfrak{G}^+_{1217434451} = 12.946690\ldots$

[8] It is called the `guru64` interface; see the user's manual of `fftw` [10].

We then looked for prime numbers $q$ such that $\nu(q) > \nu(964477901) = 1.2369344\ldots$ and we found that $\nu(2918643191) = 1.2440460\ldots$ In about 90 min of computation time for the FFT-step on the same machine mentioned before we got that $\mathfrak{G}_{2918643191} = 0.302789\ldots$ and $\mathfrak{G}^{+}_{2918643191} = 12.573983\ldots$, using the long double precision. In this case it seems that procedure in a) is much less stable than the one in b) probably because of the fact that $T(x)$ and $\psi(x)$ are much larger, for $x \to 0^{+}$, than $S(x)$ and $\log(\Gamma(x))$. Computations for further "good" candidates, in the sense that $\nu(q) > 1.18$, like $q =$ 193894451, 212634221, 251160191 538906601, 1139803271, 1217434451, 1806830951, 2488788101, 2830676081, 7079770931 were also performed. The computations for these primes were performed on the cluster previously mentioned.

Moreover, for $q = 9109334831$ we got that $\mathfrak{G}_{9109334831} = -0.248739\ldots$, thus obtaining a new minimal value for $\mathfrak{G}_q$ and a new example of Theorem 4 of [9]; at the same time we also got $\mathfrak{G}^{+}_{9109334831} = 12.128187\ldots$ The precomputations for this case, performed with the same strategy used for the smaller primes $q$ mentioned in this paragraph, required about nine days on the cluster and the FFTs computation required about 1000 min on the Xeon machine mentioned before (this amount of time also depends on a runtime RAM swapping phenomenon) or 312 min on the new CAPRI infrastructure of the University of Padova ("Calcolo ad Alte Prestazioni per la Ricerca e l'Innovazione"; whose CPU is an Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz, with 256 cores and equipped with 6TB of RAM). Such a result was then double-checked on CAPRI using the much slower algorithm a). A further new example of Theorem 4 of [9] we obtained is $\mathfrak{G}_{9854964401} = -0.096465\ldots$ which required about ten days of time for the precomputations and 326 min for the FFT stage on CAPRI. As usual the result was double-checked using the approach a).

All the results mentioned in this paragraph are collected in Table 5. The PARI/GP scripts and the C programs used and the computational results obtained are available at the following web address: http://www.math.unipd.it/~languasc/EK-comput.html.

## 5 On the absolute value of the logarithmic derivative of Dirichlet *L*-functions

Using (10)–(12), (14) and (18)–(19), for every odd prime $q$ we immediately get

$$M_q^{\mathrm{odd}} := \max_{\chi \text{ odd}} \left|\frac{L'}{L}(1,\chi)\right| = \max_{\chi \text{ odd}} \left|\gamma + \log(2\pi) + \frac{1}{B_{1,\overline{\chi}}} \sum_{a=1}^{q-1} \overline{\chi}(a) \log\left(\Gamma\left(\frac{a}{q}\right)\right)\right|$$

and

$$M_q^{\mathrm{even}} := \max_{\substack{\chi \neq \chi_0 \\ \chi \text{ even}}} \left|\frac{L'}{L}(1,\chi)\right| = \max_{\substack{\chi \neq \chi_0 \\ \chi \text{ even}}} \left|\gamma + \log(2\pi) - \frac{1}{2} \frac{\sum_{a=1}^{q-1} \overline{\chi}(a)\, S(a/q)}{\sum_{a=1}^{q-1} \overline{\chi}(a) \log\left(\Gamma(a/q)\right)}\right|.$$

Hence we can compute $M_q = \max_{\chi \neq \chi_0} |L'/L(1,\chi)| = \max(M_q^{\mathrm{odd}}, M_q^{\mathrm{even}})$ using the values of $\log\Gamma$ and $S$ obtained for the computation of $\mathfrak{G}_q$ and $\mathfrak{G}_q^{+}$. In Table 2 we give the values of $M_q$ for every odd prime up to 300 computed, using PARI/GP, with a precision of 30 digits. Using the data in Sect. 4.5.1 we also computed, on the Dell Optiplex machine previously mentioned, the values of $M_q$ and $M_q/\log\log q$ for every odd prime $q \le 10^6$ and in Figs. 3 and 4 we inserted their scatter plots that largely extend Fig. 1 of Ihara et al. [17] (please remark that our $M_q$ is denoted as $Q_m$ there). Such data in Figs. 3 and 4 also fit, for $q$ sufficiently large, with the estimate $M_q \le (2 + o(1)) \log\log q$ as $q$ tends to

infinity, proved, under the assumption of GRH, in Theorem 3 of [17]. We also remark that $M_q^{\text{odd}} > M_q^{\text{even}}$ for 62521 cases over a total number of primes equal to 78497 (79.65%) and that $M_q^{\text{even}} > M_q^{\text{odd}}$ in the remaining 15976 cases (20.35%).

The complete list of numerical results for $M_q$ can be downloaded at the following web address: https://www.math.unipd.it/~languasc/EKcomput/results.

## 6 On the generalised Euler constants in arithmetic progressions $\gamma_k(a, q)$

Recall that $q$ is an odd prime. In the case we have to precompute $T(a/q)$ and we also need $\psi(a/q)$. Hence, as a by-product we can also obtain the values of the generalised Euler constants $\gamma_0(a, q)$ and $\gamma_1(a, q)$, see Sect. 6.1–6.2. In practice this is done by activating an optional flag in the main gp script. The computation of $\gamma_k(a, q)$ for $k \geq 2$ is described in Sect. 6.3.

### 6.1 Generalised Euler constants $\gamma_0(a, q)$

For $\gamma_0(a, q)$ with $1 \leq a \leq q-1$, $q$ odd prime, by (4) we have

$$\gamma_0(a, q) = -\frac{1}{q}\Big(\log q + \psi\big(\frac{a}{q}\big)\Big).$$

Recalling that $\psi(1) = -\gamma$, we also have $\gamma_0(q, q) = (\gamma - \log q)/q$.

### 6.2 Generalised Euler constants $\gamma_1(a, q)$

For $\gamma_1(a, q)$ with $1 \leq a \leq q-1$, $q$ odd prime, we can use (4) and (7). This way we get

$$\begin{aligned}\gamma_1(a, q) &= -\frac{1}{q}\Big(\frac{(\log q)^2}{2} + (\log q)\psi\big(\frac{a}{q}\big) + \psi_1\big(\frac{a}{q}\big)\Big)\\ &= \frac{1}{q}\Big(\gamma_1 - \frac{(\log q)^2}{2} - (\log q)\psi\big(\frac{a}{q}\big) - T\big(\frac{a}{q}\big)\Big).\end{aligned}$$

Moreover, since $\psi(1) = -\gamma$ and $T(1) = 0$, we also have

$$\gamma_1(q, q) = \frac{1}{q}\Big(\gamma_1 + \gamma\log q - \frac{(\log q)^2}{2}\Big).$$

Using the formulae in the previous two paragraphs we computed $\gamma_0(a, q)$ and $\gamma_1(a, q)$ with $q$ prime, $3 \leq q \leq 100$, $1 \leq a \leq q$, in about 4 s of computation time with a precision of 30 digits.

Such results are listed towards the end of the gp-script file that can be downloaded here: http://www.math.unipd.it/~languasc/EK-comput.html.

### 6.3 The general case $\gamma_k(a, q)$, $k \geq 2$

The general case $\gamma_k(a, q)$, $k \in \mathbb{N}$, $k \geq 2$, $q \geq 1$, $1 \leq a \leq q$, do not follow from the data already computed for the Euler–Kronecker constants since we need information about the values of $\psi_n(x)$, for every $2 \leq n \leq k$. Such a direct computation of both $\psi_n(a/q)$ and $\gamma_n$ can be easily performed via Eqs. (4)–(5) using the PARI/GP summing function `sumnum` paying attention to submit a sufficiently fast convergent sum. For example, to compute $\gamma_n$, $n \in \mathbb{N}$, we used the formulae

$$\gamma_n = \sum_{m=1}^{+\infty}\Big(\frac{(\log m)^n}{m} - \frac{1}{n+1}\sum_{j=0}^{n}\binom{n+1}{j}(\log m)^j\big(\log\big(1+\frac{1}{m}\big)\big)^{n+1-j}\Big) \tag{31}$$

**Table 6 Computation of the generalised Euler constants $\gamma_n$, $0 \le n \le 30$, with a precision of at least 40 digits; computed with PARI/GP, v. 2.11.4**

| $n$ | $\gamma_n$ |
|---|---|
| 0 | 0.57721566490153286060651209008240243 10 ... |
| 1 | −0.07281584548367672486058637587490131 91 ... |
| 2 | −0.00969036319287231848453038603521252 93 ... |
| 3 | 0.00205383442030334586616004654275338 42 ... |
| 4 | 0.00232537006546730005746817017752606 80 ... |
| 5 | 0.00079332381730106270175333487744444 48 ... |
| 6 | −0.00023876934543019960987242184190800 42 ... |
| 7 | −0.00052728956705775104607409750547885 82 ... |
| 8 | −0.00035212335380303950960205216500120 87 ... |
| 9 | −0.00003439477441808804817791462379822 73 ... |
| 10 | 0.00020533281490906479468372228923706 53 ... |
| 11 | 0.00002701844395439035266729020820679556 ... |
| 12 | 0.00016727291210514019335350154334118 34 ... |
| 13 | −0.00002746380660376015886000760369335 51 ... |
| 14 | −0.00020920926205929994583713969734458 49 ... |
| 15 | −0.00028346865532024144664293447499712 69 ... |
| 16 | −0.00019969685830896977470778456320324 03 ... |
| 17 | 0.00002627703710991833669946659763051 01 ... |
| 18 | 0.00030736840814925282659275475194862 56 ... |
| 19 | 0.00050360545304735562905559643771716 00 ... |
| 20 | 0.00046634356151155944940059482443355 05 ... |
| 21 | 0.00010443776975600011581079567436772 04 ... |
| 22 | −0.00054159958220399770165519617317410 55 ... |
| 23 | −0.00124396209040824577929974159953716 58 ... |
| 24 | −0.00158851127890356156190619661152111 58 ... |
| 25 | −0.00107459195273848882472429198735317 30 ... |
| 26 | 0.00065680351863715443150477300335621 52 ... |
| 27 | 0.00347783691361853820900735957425881 15 ... |
| 28 | 0.00640006853170062945810722822194586 36 ... |
| 29 | 0.00737115177047223913441240242355940 21 ... |
| 30 | 0.00355772885557316094791353774890840 26 ... |

and

$$\gamma_n = \sum_{m=1}^{+\infty} \left( (\log m)^n \left( \frac{1}{m} - \log\left(1 + \frac{1}{m}\right) \right) - \frac{1}{n+1} \sum_{j=0}^{n-1} \binom{n+1}{j} (\log m)^j \left(\log\left(1 + \frac{1}{m}\right)\right)^{n+1-j} \right), \tag{32}$$

which both easily follow from (6). We get, in less than 7 s of time and with a precision of at least 40 digits, the results in Table 6; to be sure about the correctness of such results we computed them twice using the formulae (31)–(32) and then we compared the outcomes. These values are in agreement with the data on p. 282 of Bohman–Fröberg [3] for $n = 0, \ldots, 20$. For larger $n$'s the formulae in (31)–(32) seem to be not good enough to get precise results via the `sumnum` function with this precision level.

To compute $\psi_n(a/q)$ and, as a consequence, $\gamma_k(a, q)$, we can proceed in a similar way as we did for $T(a/q)$ and $\gamma_1(a, q)$, see the program `Gen-Euler-constants.gp` here http://www.math.unipd.it/~languasc/EK-comput.html. Towards the end of this program

file you can find a large list (too long to be included here) of computed values of $\gamma_k(a,q)$ for $1 \le k \le 20$, $1 \le q \le 9$, $1 \le a \le q$, with a precision of 20 digits. In about 50 s of computation time we replicated Dilcher's computations, on pp. S21–S24 of [7].

**Authors' contributions**
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", http://capri.dei.unipd.it. I also wish to thank Karim Belabas and Bill Allombert (University of Bordeaux) for a couple of key suggestions about `libpari` and `gp2c` and Luca Righi (University of Padova) for his help in developing the quadruple precision versions of the fft-programs, in designing the parallelised precomputations and in organising the use of the cluster of the Dipartimento di Matematica "Tullio Levi-Civita", http://computing.math.unipd.it/highpc, and the use of CAPRI.
Finally, I warmly thank the referee for his/her suggestions and remarks.

**Funding** Open access funding provided by Universitá degli Studi di Padova within the CRUI-CARE Agreement.

Received: 13 November 2019 Accepted: 3 September 2020
Published online: 25 December 2020

**References**

1. Berger, A.: Sur une sommation des quelques séries. Nova Acta Reg. Soc. Sci. Ups. **12**, 29–31 (1883)
2. Berndt, B.C.: Ramanujan's Notebooks. Part I. Springer, Berlin (1985)
3. Bohman, J., Fröberg, C.-E.: The Stieltjes function-definition and properties. Math. Comp. **51**, 281–289 (1988)
4. Cohen, H.: Number Theory. Volume I: Tools and Diophantine Equations. Graduate Texts in Mathematics, vol. 239. Springer, Berlin (2007)
5. Cohen, H.: Number Theory. Volume II: Analytic and Modern Tools. Graduate Texts in Mathematics, vol. 240. Springer, New York (2007)
6. Deninger, C.: On the analogue of the formula of Chowla and Selberg for real quadratic fields. J. Reine Angew. Math. **351**, 171–191 (1984)
7. Dilcher, K.: Generalized Euler constants for arithmetical progressions. Math. Comp. **59**, 259–282 (1992)
8. Dilcher, K.: On generalized gamma functions related to the Laurent coefficients of the Riemann zeta function. Aequationes Math. **48**, 55–85 (1994)
9. Ford, K., Luca, F., Moree, P.: Values of the Euler $\phi$-function not divisible by a given odd prime, and the distribution of Euler-Kronecker constants for cyclotomic fields. Math. Comp. **83**, 1447–1476 (2014)
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 http://www.fftw.org
11. Gnu Scientific Library, version `2.5`, 2018. http://www.gnu.org/software/gsl/
12. Gun, S., Murty, M.R., Rath, P.: Transcendental nature of special values of $L$-functions. Can. J. Math. **63**, 136–152 (2011)
13. Gut, M.: Die Zetafunktion, die Klassenzahl und die Kronecker'sche Grenzformel eines beliebigen Kreiskorpers. Comment. Math. Helv. **1**, 160–226 (1929)
14. Kanemitsu, S.: On evaluation of certain limits in closed form. In: de Koninck, Levesque (eds.) Théorie des nombres, Proceedings of the International Number Theory Conference, Université Laval, July 5–18, 1987, De Gruyter 1989, pp. 459–474
15. Ihara, Y.: The Euler-Kronecker invariants in various families of global fields. In: Ginzburg, V. (ed.) Algebraic Geometry and Number Theory. In Honor of Vladimir Drinfeld's 50th Birthday, Progress in Mathematics, vol. 850, pp. 407–451. Birkhäuser Boston, Cambridge, MA (2006)
16. Ihara, Y.: On "$M$-functions" closely related to the distribution of $L'/L$-values. Publ. Res. Inst. Math. Sci. **44**, 893–954 (2008)
17. Ihara, Y., Murty, V.K., Shimura, M.: On the logarithmic derivatives of Dirichlet $L$-functions at $s = 1$. Acta Arith. **137**, 253–276 (2009)
18. Katayama, K.: Class number formulas, Kronecker's limit formulas, Chowla–Selberg formulas and the generalized gamma functions. J. Number Theory **133**, 2092–2120 (2013)
19. Languasco, A., Moree, P., Saad Eddin, S., Sedunova, A.: Computation of the Kummer ratio of the class number for prime cyclotomic fields, arxiv (2019), http://arxiv.org/abs/1908.01152
20. Moree, P.: Irregular Behaviour of Class Numbers and Euler-Kronecker Constants of Cyclotomic Fields: The Log Log Log Devil at Play, Irregularities in the Distribution of Prime Numbers. In: Pintz, J., Rassias, MTh (eds.) From the Era of Helmut Maier's Matrix Method and Beyond, pp. 143–163. Berlin, Springer (2018)
21. The PARI Group, PARI/GP version `2.11.4`, Bordeaux, 2020. http://pari.math.u-bordeaux.fr/
22. Rader, C.M.: Discrete Fourier transforms when the number of data samples is prime. Proc. IEEE **56**, 1107–1108 (1968)

## Publisher's Note