RAPID COMPUTATION OF SPECIAL VALUES OF DIRICHLET L-FUNCTIONS

FREDRIK JOHANSSON

ABSTRACT. We consider computing the Riemann zeta function $\zeta(s)$ and Dirichlet L-functions $L(s,\chi)$ to p-bit accuracy for large p. Using the approximate functional equation together with asymptotically fast computation of the incomplete gamma function, we observe that $p^{3/2+o(1)}$ bit complexity can be achieved if s is an algebraic number of fixed degree and with algebraic height bounded by $O(p)$. This is an improvement over the $p^{2+o(1)}$ complexity of previously published algorithms and yields, among other things, $p^{3/2+o(1)}$ complexity algorithms for Stieltjes constants and $n^{3/2+o(1)}$ complexity algorithms for computing the nth Bernoulli number or the nth Euler number exactly.

1. INTRODUCTION

Let χ be a Dirichlet character modulo $q \geq 1$. The associated Dirichlet L-function is the analytic continuation of

$$L(s,\chi) = \sum_{n=1} \frac{\chi(n)}{n^s}, \qquad \operatorname{Re}(s) > 1 \tag{1}$$

to $s \in \mathbb{C}$ with the possible exception of a pole at $s = 1$. The Riemann zeta function $\zeta(s)$ is the Dirichlet L-function corresponding to the trivial character $\chi(n) = 1$, which is the unique character modulo $q = 1$.

If χ is a primitive character, then the function $L(s,\chi)$ is represented in the entire complex plane by a convergent expansion, the *approximate functional equation* [Coh19, Theorem 7.3]

$$\begin{aligned}\Gamma\left(\frac{s+\delta}{2}\right) L(s,\chi) = \delta_{q,1}\pi^{s/2}\left(\frac{\alpha^{(s-1)/2}}{s-1} - \frac{\alpha^{s/2}}{s}\right) + \sum_{n=1}^{\infty} \frac{\chi(n)}{n^s}\,\Gamma\left(\frac{s+\delta}{2}, \frac{\pi n^2 \alpha}{q}\right) \\ + \omega\left(\frac{\pi}{q}\right)^{s-1/2} \sum_{n=1}^{\infty} \frac{\overline{\chi}(n)}{n^{1-s}}\,\Gamma\left(\frac{1-s+\delta}{2}, \frac{\pi n^2}{\alpha q}\right)\end{aligned} \tag{2}$$

where $\delta \in \{0,1\}$ is the parity $\chi(-1) = (-1)^\delta$, ω is the *root number* of χ, which satisfies $|\omega| \leq 1$, and $\Gamma(a,z) = \int_z^\infty t^{a-1}e^{-t}dt$ is the incomplete gamma function. The quantity α is a free positive parameter; we may take $\alpha = 1$ to balance the rate of convergence of both series. If χ is not primitive, we can decompose $L(s,\chi)$ in terms of primitive functions.

The expansion (2) is useful for high-precision computation of $L(s,\chi)$ due to the super-exponential decay $\Gamma(a, Cn^2) \approx \exp(-Cn^2)$ of the incomplete gamma functions. We need only $O(p^{1/2})$ terms for a desired bit precision p, which should be contrasted with Euler-Maclaurin summation [BC21, §4.2] [Joh14b] and similar

2020 *Mathematics Subject Classification.* Primary 33F05, 11M06; Secondary 33B20, 65D20.

methods which require $O(p)$ terms. (An equally important advantage of (2) is that we only need $O(q^{1/2})$ terms as a function of the modulus q.)

The drawback of (2) is that we have to compute the nonelementary incomplete gamma functions. Our goal is to study this problem with attention to the bit complexity when $p \to \infty$. We use "time" and "bit operations" synonymously, and recall that floating-point numbers with p-bit precision can be multiplied in time $O(p \log p) = p^{1+o(1)}$ [HvdH21]. Using Euler-Maclaurin summation, for example, it is easy to show that we can compute $L(s, \chi)$ or any of its s-derivatives $L^{(j)}(s, \chi)$ to p-bit accuracy in time $p^{2+o(1)}$ for fixed s, χ and j. Our main observation is the following improved complexity bound for special values s.

Theorem 1. *Let χ be a fixed Dirichlet character, and let $s \in \overline{\mathbb{Q}}$ be an algebraic number of fixed degree such that the minimal polynomial of s over $\mathbb{Z}$ has coefficients bounded in absolute value by $O(p)$. Then, for any fixed $j \geq 0$, the value $L^{(j)}(s, \chi)$ can be approximated with absolute error less than 2^{-p} in time $p^{3/2+o(1)}$ using $p^{1+o(1)}$ space. (When $s = 1$ and this point is a pole, the corresponding Laurent series coefficient is computed.)*

Proof. The height condition implies that $|s| = O(p)$. Since any terms and prefactors appearing in (2) and in the asymptotics of the incomplete gamma function are bounded by $\exp(|s|^{1+o(1)})$, it is sufficient to truncate both infinite series to $N = p^{1/2+o(1)}$ terms and approximate the terms to $p^{1+o(1)}$ bits.

The function $y(z) = \Gamma(a, z)$ is holonomic, satisfying a linear differential equation $Ay = 0$ with $A \in \mathbb{Z}[a, z, \frac{d}{dz}]$. We evaluate y (for two values of the parameter a) at N points z. We can compute each such function value in $p^{1+o(1)}$ bit operations and $p^{1+o(1)}$ space using the *bit-burst algorithm* [CC90], employing arithmetic in the number field $\mathbb{Q}(s)$. This bound holds uniformly for the required values of a and z, by the same argument as in [Mez12, Corollary 1], using the facts that $|z| \leq p^{1+o(1)}$ and that A as well as the defining polynomial of $\mathbb{Q}(s)$ have fixed degree and coefficients of bit size $O(\log p)$.

Using standard methods, the remaining operations (evaluation of Dirichlet characters, the gamma function, and elementary functions) fall within the same complexity bounds.

For derivatives $L^{(j)}(s, \chi)$, and at removable singularities, the equivalent operations can be carried out using arithmetic on truncated formal power series. □

We will provide additional details below. The only interesting point in the proof of Theorem 1 is the use of the bit-burst algorithm instead of naive summation, which allows us to compute the function $\Gamma(a, z)$ in quasilinear rather than quadratic time. The bit-burst algorithm for holonomic functions has been known since the 1980s [CC90, vdH99, vdH01, Mez11, Mez12], and since the 1970s in special cases [Bre76a], yet we are not aware of a correct complexity bound of this kind in the literature for Dirichlet L-functions or even for the special case of the Riemann zeta function.

The use of the approximate functional equation for L-function computation has been studied in detail by Rubinstein [Rub98] and several other authors [Dok04, Boo06, Mol10, BC21]. These works do not mention the bit-burst algorithm or address the bit complexity for large p, instead focusing on parameters relevant for numerical testing of the generalized Riemann hypothesis, namely with fixed p and with large $|\operatorname{Im}(s)|$ and/or large q. For large $|\operatorname{Im}(s)|$, one must use a "smoothed"

version of (2) to avoid exponentially large cancellation, or Riemann-Siegel type expansions; the method in Theorem 1 is not competitive in this setting, where the best methods achieve $O(|\operatorname{Im}(s)|^{1/2})$ or lower complexity for a fixed level of accuracy.

Borwein, Bradley and Crandall [BBC00] and Crandall [Cra12, BB15] discuss the approximate functional equation in the context of high-precision zeta function computation, but do not mention the bit-burst algorithm or give a complexity bound of this type. Crandall [Cra12] writes that the incomplete gamma function can be computed using $p^{1+o(1)}$ "operations", but this is referring to full-precision arithmetic operations, which would give us $p^{2+o(1)}$ bit complexity for $\Gamma(a, z)$ and $p^{5/2+o(1)}$ bit complexity for L-functions. A refined algorithm that achieves $p^{2+o(1)}$ bit complexity is described in [BBC00, §7]; see §5 below.

In a 1988 paper, Borwein and Borwein [BB88] claim that $\zeta(s)$ can be computed in time $p^{1+o(1)}$ if s is a fixed rational number, and in time $p^{3/2+o(1)}$ if s is a fixed generic (computable) complex number. No explicit algorithm is given to justify these claims: the authors simply write "we truncate both the integral and the sum" with reference to the formula

$$\zeta(s)\Gamma\left(\frac{s}{2}\right)\pi^{-s/2} - \frac{1}{s(s-1)} = \int_1^\infty \frac{t^{(1-s)/2} + t^{s/2}}{t} \sum_{n=1}^{\infty} e^{-n^2\pi t}\, dt \tag{3}$$

which, apart from minor differences in notation, is the approximate functional equation for $\zeta(s)$ in the form originally derived by Riemann [Rie59] (we obtain the series in incomplete gamma functions by integrating term by term).

Both bounds claimed by the Borweins are a factor $p^{1/2}$ better than all methods known to this author. Lacking evidence to the contrary, we believe that the Borweins had in mind some combination of the algorithms that will be described below and that their complexity analysis was erroneous. Our goal with this article is therefore in part to correct the record.[1]

The rest of this paper is structured as follows: §2 discusses some consequences of Theorem 1, §3 gives a more detailed description of the algorithm, and §4 reports implementation results. Finally, §5 discusses alternative algorithms for use when s is not algebraic.

2. Applications and generalizations

The immediate application of Theorem 1 is that $p^{3/2+o(1)}$ can be a significant improvement over $p^{2+o(1)}$ for numerical evaluation to tens of thousands of digits. Such computations are not exclusively done to test algorithms; for example, integer relation searches employing 50,000-digit precision have been successful in discovering new identities involving special values of L-functions [BB01].

2.1. Values at integers.

The most famous special values, $\zeta(n)$ and $L(n, \chi)$ with $n \in \mathbb{Z}$, can be expressed in terms of logarithmic derivatives of the gamma function at rational points, and can consequently be computed in quasilinear time $p^{1+o(1)}$ using binary splitting [Kar98, Joh21].

[1] We mention that J. Borwein coauthored the 2000 survey paper [BBC00] on $\zeta(s)$ computation, which discusses the approximate functional equation prominently but does not mention the claims from 1988. This omission suggests that the Borweins were aware of the error (but perhaps did not consider it important enough to publish a correction). The other complexity results in [BB88], for instance concerning $\Gamma(s)$, are correct.

Alternatively (and often more efficiently), binary splitting can be applied directly to convergence-accelerated series for the Riemann zeta function [Joh14a, §4.7] or hypergeometric series for particular values such as

$$\zeta(3) = \frac{5}{2}\sum_{n=1}^{\infty}(-1)^{n+1}\frac{(n!)^2}{n^3(2n)!} \tag{4}$$

which have been used to compute billions of digits [SG03, Yee21].

However, these quasilinearity results all assume that $n = p^{o(1)}$. For example, if n and p are proportional (or proportional up to logarithmic factors), then the complexity degenerates to $p^{2+o(1)}$ or worse. The complexity is also softly quadratic with $n \propto p$ if we compute $L(s, \chi)$ directly using the L-series (1) or the corresponding Euler product. The $p^{3/2+o(1)}$ complexity of Theorem 1 is then an improvement over previous algorithms.

2.2. **Bernoulli and Euler numbers.** The Bernoulli numbers and Euler numbers are the rational numbers and integers respectively defined by

$$\frac{x}{e^x - 1} = \sum_{n=0}^{\infty}\frac{B_n}{n!}x^n, \quad \frac{1}{\cosh(x)} = \sum_{n=0}^{\infty}\frac{E_n}{n!}x^n. \tag{5}$$

The odd-index values are trivial, while the even-index values can be expressed in terms of Dirichlet L-functions as

$$B_{2n} = (-1)^{n+1}\frac{2(2n)!}{(2\pi)^{2n}}\zeta(2n), \quad E_{2n} = (-1)^n\frac{4^{n+1}(2n)!}{\pi^{2n+1}}\beta(2n+1) \tag{6}$$

where $\beta(s) = L(s, \chi_{4.3})$ is the Dirichlet beta function, corresponding to the character modulo $q = 4$ with $(\chi_{4.3}(n))_{n=0}^{\infty} = (0, 1, 0, -1, \ldots)$.

There are $\Theta(n \log n)$ bits in E_n and in the numerator of B_n (the denominator is easy to determine), so we can recover the exact values by evaluating the L-functions numerically to $p = n^{1+o(1)}$ bits. As a corollary of Theorem 1, we have the following:

Theorem 2. *The nth Bernoulli number B_n and Euler number E_n can be computed exactly in time $n^{3/2+o(1)}$ using $n^{1+o(1)}$ space.*

The significance of this result is that all methods known until quite recently (for instance those employing the Euler product) require at least $n^{2+o(1)}$ time.

Harvey [Har14] gave the first subquadratic algorithm for computing B_n, which uses $n^{4/3+o(1)}$ time and space or $n^{3/2+o(1)}$ time when confined to $n^{1+o(1)}$ space. We fail to improve on Harvey's bound, but the methods are independent: ours is numerical; Harvey's uses modular arithmetic and does not involve L-functions. For Euler numbers, no subquadratic algorithm has been published before ours, though it is plausible that Harvey's algorithm can be generalized to this case.

2.3. **Sparse zeta-expansions.** Many slowly converging series or products can be evaluated to high precision using *zeta function acceleration* [FV96], which is based on the formal rearrangement

$$\sum_{n\in A} f(1/n) = \sum_{m} f_m \zeta_A(m), \quad f(z) = \sum_{m} f_m z^m, \ \zeta_A(s) = \sum_{n\in A} n^{-s}. \tag{7}$$

The complexity of approximating such a sum to p-bit accuracy is typically $p^{2+o(1)}$ provided that the transformed series converges geometrically and that $O(p)$ coefficients f_m and zeta values $\zeta_A(m)$ can be evaluated simultaneously to p-bit accuracy

in time $p^{2+o(1)}$ using FFT-based power series operations (this is the case if f is elementary and $\zeta_A(s) = \zeta(s)$, for example).

If the resulting zeta-expansion is sparse, then Theorem 1 may yield an improved complexity bound. An example is the Landau-Ramanujan constant

$$\lambda = \left(\frac{1}{2} \prod_{p \equiv 3 \bmod 4} \frac{1}{1-p^{-2}} \right)^{1/2} \approx 0.764 \tag{8}$$

which appears in the asymptotic formula $\lambda x / \sqrt{\log(x)}$ for the number of integers $k \leq x$ expressible as a sum of two squares. Flajolet and Vardi [FV96] obtain the sparse zeta-type expansion

$$\lambda = \frac{1}{\sqrt{2}} \prod_{n=1}^{\infty} \left[\left(1 - \frac{1}{2^{2^n}}\right) \frac{\zeta(2^n)}{\beta(2^n)} \right]^{1/2^{n+1}} \tag{9}$$

which requires only $O(\log(p))$ terms for p-bit accuracy. It follows from Theorem 1 that we can compute λ to p-bit accuracy in time $p^{3/2+o(1)}$.

2.4. **Stieltjes constants.** The Stieltjes constants γ_k are, up to a scaling factor, the coefficients in the Laurent series of $\zeta(s)$ at $s = 1$. Theorem 1 states that we can compute any Stieltjes constant to p-bit accuracy in time $p^{3/2+o(1)}$, which again is superior to the $p^{2+o(1)}$ complexity of classical methods like Euler-Maclaurin summation [LT72, Joh14b] as well as methods based on numerical integration [JB18]. Algorithms with $p^{1+o(1)}$ complexity are only known for Euler's constant $\gamma = \gamma_0$ [BM80].

The idea of using (2) to compute Stieltjes constants is of course not new. Coffey [Cof14, Proposition 9] gives the explicit formula

$$\gamma = \log(4\pi) - 2 + \frac{2}{\sqrt{\pi}} \sum_{n=1}^{\infty} \frac{1}{n} \Gamma(\tfrac{1}{2}, \pi n^2) + 2 \sum_{n=1}^{\infty} \Gamma(0, \pi n^2) \tag{10}$$

along with a much more complex formula for γ_1 written in terms of series of ${}_2F_2$ and ${}_3F_3$ hypergeometric functions and digamma functions. Coffey notes that these formulas "may have some attraction for computation" due to the $e^{-\pi n^2}$ type decrease of the terms. He also notes that the method can be generalized to Dirichlet L-function analogs of Stieltjes constants. As indicated in the proof of Theorem 1, implementing (2) with power series arithmetic provides such a generalization without requiring the derivation of unwieldy formulas for the higher derivatives.

Similarly, we obtain a $p^{3/2+o(1)}$ complexity algorithm for the Glaisher–Kinkelin constant $e^{1/2-\zeta'(-1)}$, Keiper-Li coefficients, etc.

The approximate functional equation has been used in a somewhat different way by Keiper [Kei92] to compute Stieltjes constants and other series coefficients related to the Riemann zeta function, with worse complexity than Theorem 1; we revisit this topic in §5.

2.5. **Values at rational points.** Theorem 1 implies $p^{3/2+o(1)}$ complexity for computing the values $L(s, \chi)$ with $s \in \mathbb{Q}$. These constants have various applications; $\zeta(n + 1/2)$ and $\beta(n + 1/2)$ appear in thermodynamics (Bose-Einstein statistics) and in connection with lattice sums describing the electrostatic potentials in crystals (Madelung constants) [Fin03, §1.10]. It is a famous open problem whether $L(1/2, \chi) \neq 0$ for all primitive characters χ [Pla11, §7.6].

The number $\zeta(1/2) \approx -1.4603545$ makes an interesting appearance in a formula in Ramanujan's lost notebook (see [AB13, §8.3], where generalizations to other values of $\zeta(s)$ and $L(s,\chi)$ with $s \in \mathbb{Q}$ are discussed as well). For any $\alpha, \beta > 0$ such that $\alpha\beta = 4\pi^3$,

$$\sum_{n=1}^{\infty} \frac{1}{e^{n^2\alpha} - 1} = \frac{\pi^2}{6\alpha} + \frac{1}{4} + \frac{\sqrt{\beta}}{4\pi}\left[\zeta(\tfrac{1}{2}) + \sum_{n=1}^{\infty} \frac{\cos(\sqrt{n\beta}) - \sin(\sqrt{n\beta}) - e^{-\sqrt{n\beta}}}{\sqrt{n}(\cosh(\sqrt{n\beta}) - \cos(\sqrt{n\beta})}\right]. \quad (11)$$

There is a parallel to the free parameter in (2): by varying α, we can force either the left or the right series to converge faster at the expense of the other. Setting $\alpha = O(1/p)$ in Ramanujan's formula minimizes the total number of terms for a given precision p; this leads to an algorithm with $p^{2+o(1)}$ bit complexity to compute $\zeta(1/2)$, comparable to Euler-Maclaurin summation and inferior to Theorem 1.

2.6. **Hurwitz zeta-type functions.** The method behind Theorem 1 is not restricted to "proper" L-functions. Crandall [Cra12, BB15] has given a formula analogous to (2) (Crandall calls this a "Riemann-splitting representation") for the Lerch transcendent, which is the analytic continuation of the series

$$\Phi(z, s, a) = \sum_{n=0}^{\infty} \frac{z^n}{(n+a)^s}, \quad |z| < 1. \quad (12)$$

Combining Crandall's expansion with bit-burst evaluation of the incomplete gamma function should lead to $p^{3/2+o(1)}$ algorithms for the following:

- The Lerch transcendent $\Phi(z, s, a)$ with $s \in \overline{\mathbb{Q}}$, $z, a \in \mathbb{C}$ and its s-derivatives.
- The Hurwitz zeta function $\zeta(s, a)$ with $s \in \overline{\mathbb{Q}}, a \in \mathbb{C}$ and its s-derivatives.
- The generalized Stieltjes constants $\gamma_n(a)$ with $a \in \mathbb{C}$.
- The polylogarithm $\mathrm{Li}_s(z)$ with $s \in \overline{\mathbb{Q}}, z \in \mathbb{C}$ and its s-derivatives.
- The Barnes G-function $G(z)$ with $z \in \mathbb{C}$.

We have not checked the details of these computations (validity of analytic continuations, possible exceptional points, explicit error bounds, uniform complexity with respect to parameters), and we leave this for a future study.

As in the case of integer zeta values, $p^{1+o(1)}$ algorithms are already available for the above functions in some more restricted cases, e.g. for $\zeta(n, a)$ with $a \in \mathbb{Q}$.

3. The algorithm

Since the proof of Theorem 1 above is quite terse and the bit-burst algorithm for generic holonomic functions requires much more complicated machinery than in the specialized case of computing $\Gamma(a, z)$, we give a more explicit description here.

We may rely on ball arithmetic [vdH09, Joh17], which means that explicit error bounds need to be derived only for the truncation errors in infinite series; asymptotic estimates suffice for choosing the floating-point precision.

3.1. **The outer series.** To bound the tails of the infinite series in (2), the following formulas may be used. Similar bounds can also be found in [Rub98].

Lemma 3. *For real $z > 0$ and complex $a = \sigma + \tau i$, the order $j \geq 0$ parameter derivative of the incomplete gamma function $\Gamma(a, z)$ satisfies the bound*

$$\begin{aligned}\left|\Gamma^{(j,0)}(a,z)\right| &= \left|z^{a-1}\log^j(z)e^{-z}\int_0^\infty e^{-t}\left(1+\frac{t}{z}\right)^{a-1}\left(1+\frac{\log(1+t/z)}{\log(z)}\right)^j dt\right| \\ &\leq z^{\sigma-1}\log^j(z)e^{-z}\int_0^\infty \exp\left(-t+\frac{(\sigma-1)t}{z}+\frac{jt}{z\log(z)}\right)dt \\ &\leq \frac{z^{\sigma-1}\log^j(z)e^{-z}}{1-B_j(\sigma,z)}, \quad B_j(\sigma,z) = \frac{1}{z}\left(\max(\sigma-1,0)+\frac{j}{\log(z)}\right)\end{aligned} \tag{13}$$

provided that $B_j(\sigma, z) < 1$, and assuming that $z > 1$ if $j \geq 1$.

We illustrate how to bound the zeroth derivative of the first of the two infinite series in (2):

Lemma 4. *Assume that $N, q \geq 1$, $s = \sigma + \tau i$ with $\sigma, \tau \in \mathbb{R}$, $\delta \in \{0, 1\}$. Define $C = (\sigma + \delta)/2$ and $D = \pi\alpha/q$. If $DN^2 > C - 1$, then*

$$\begin{aligned}\left|\sum_{n=N}^\infty \frac{\chi(n)}{n^s}\Gamma\left(\frac{s+\delta}{2}, \frac{\pi n^2\alpha}{q}\right)\right| &\leq \sum_{n=N}^\infty \frac{1}{n^\sigma}\frac{(Dn^2)^{C-1}e^{-Dn^2}}{1-B_0(C,Dn^2)} \\ &= \frac{D^{C-1}}{1-B_0(C,DN^2)}\sum_{n=N}^\infty \frac{e^{-Dn^2}}{n^{2-\delta}} \\ &\leq \frac{D^{C-1}}{1-B_0(C,DN^2)}\frac{e^{-DN^2}}{N^{2-\delta}(1-e^{-D})}.\end{aligned} \tag{14}$$

We can bound the tails of the s-derivatives as follows: we expand the power series product $n^{-(s+X)}\Gamma((s+X+\delta)/2, \pi\alpha n^2/q)$ symbolically, apply the bound (13) for each coefficient, and compute geometric series bounds similar to those in (14).

The bound for the other series in (2) is identical but with $C = (1-\sigma+\delta)/2$ and $D = \pi/(q\alpha)$.

3.1.1. *Implementation remarks.* There is no need to derive a closed formula for choosing N; we can simply evaluate the bound (14) for $N = 1, 2, \ldots$ and stop when the error meets a target tolerance.

For optimal performance, we should compute a tight estimate of the number of bits that each term contributes to the final result and only compute to that precision locally. It is useful to note that for $a \in \mathbb{R}$ and $z > 0$,

$$\log(\Gamma(a,z)) \approx \begin{cases}(a-1)\log(z) - z & a < z \\ a(\log(a)-1) & a \geq z.\end{cases} \tag{15}$$

gives an accurate order-of-magnitude estimate of the incomplete gamma function.

3.2. **Evaluation of the incomplete gamma function.** The idea of the bit-burst algorithm is to analytically continue a holonomic function y using the Taylor series method for ODEs, following a path

$$z_{\text{initial}} \rightsquigarrow z_1 \rightsquigarrow z_2 \rightsquigarrow \ldots \tag{16}$$

that approaches the target point z *exponentially* and thus converges in $O(\log(p))$ steps. For example, for our application we may choose $z_1 = \lfloor z/2^{32}\rfloor 2^{32}$, $z_2 =$

$\lfloor z/2^{64} \rfloor 2^{64}$, $z_3 = \lfloor z/2^{128} \rfloor 2^{128}$, ..., where successive steps double the number of leading bits extracted from the binary expansion of z.[2] At each step, the Taylor series can be evaluated using binary splitting, and the exponentially converging steps balance the bit sizes of the coefficients against the number of terms in each Taylor series so that the overall bit complexity is quasilinear in p.

3.2.1. *Hypergeometric series.* Let $(a)_n = a(a+1)\cdots(a+n-1)$. For the first Taylor step $z_{\text{initial}} \rightsquigarrow z_1$, we may choose $z_{\text{initial}} = 0$. Here we have the hypergeometric series

$$\Gamma(a,x) = \Gamma(a) - \frac{x^a e^{-x}}{a} \sum_{n=0}^{\infty} \frac{x^n}{(a+1)_n}, \quad x = z_1, \tag{17}$$

which is valid for all $x > 0$ when $a \notin \{0, -1, -2, \ldots\}$. If the series is truncated after N terms where $N > -\operatorname{Re}(a) - 1$ and $|a + N + 1| > |x|$, then

$$\left| \sum_{n=N}^{\infty} \frac{x^n}{(a+1)_n} \right| \le \frac{|x|^N}{|(a+1)_N|} \frac{1}{1-C}, \quad C = \frac{|x|}{|a+N+1|} \tag{18}$$

Truncation bounds for derivatives of this series with respect to a can be obtained similarly; see [Joh19, Theorem 1].

At the poles of the gamma function, a limit computation is needed; this can be done using power series arithmetic or explicitly using the formula [Nat13, 8.4.15]

$$\Gamma(-n,x) = \frac{(-1)^n}{n!}(\psi(n+1) - \log(x)) - x^{-n} \left(\sum_{k=0}^{n-1} + \sum_{k=n+1}^{\infty} \right) \frac{(-x)^k}{k!(k-n)}. \tag{19}$$

When $x = z_1 \approx z$ is sufficiently large, we can also start from $z_{\text{initial}} = \infty$ and use the asymptotic series

$$\Gamma(a,x) = x^{a-1} e^{-x} \left[\sum_{n=0}^{N-1} \frac{(1-a)_n}{(-x)^n} + R_N(a,x) \right] \tag{20}$$

for the first step. We do not need (20) in the proof of Theorem 1, but practically speaking it makes a significant difference for efficiency to choose this expansion whenever $\min_N |R_N(a,x)|$ is smaller than the target tolerance. The error term satisfies $|R_N(a,x)| \le |(1-a)_N|/|x|^N$ if $a \in \mathbb{R}$ and $x > 0$ provided that $N \ge a - 1$. For error bounds with complex a, the formulas in [Nat13, §13.7] may be used.

3.2.2. *Expansions at generic points.* For consecutive Taylor steps $z_{k-1} \rightsquigarrow z_k$, we write the expansion as

$$\Gamma(a, z_k) = \sum_{n=0}^{\infty} c_n(z_{k-1}) x^n, \quad x = z_k - z_{k-1}. \tag{21}$$

where the coefficients $c_n(z_{k-1})$ need to be determined. We denote the local expansion point by u instead of z_{k-1} below to simplify the formulas.

The function $y(z) = \Gamma(a,z)$ satisfies the second-order differential equation $zy'' + (z - a + 1)y = 0$. We can translate this differential equation to the point u and convert it to the second-order linear recurrence relation

$$u(n+1)(n+2)c_{n+2}(u) + (n+1)(n+1+u-a)c_{n+1}(u) + nc_n(u) = 0. \tag{22}$$

[2]The initial number of bits is a tuning parameter; instead of the constant 32, we may start with $O(\log p)$ bits, for example.

We can also apply this recurrence to parameter derivatives. Explicitly, define

$$c_{j,n}(u) = \frac{\Gamma^{(j,n)}(a,u)}{j!n!} \tag{23}$$

so that $\Gamma^{(j,0)}(a,z_k) = \sum_{n=0}^{\infty} c_{j,n}(u)x^n$, and let $S_n = \sum_{j=0}^{J} c_{j,n}(u)X^j$ in the ring of truncated formal power series $\mathbb{C}[[X]]/\langle X^{J+1}\rangle$. Then

$$u(n+1)(n+2)S_{n+2} + (n+1)(n+1+u-a-X)S_{n+1} + nS_n = 0. \tag{24}$$

Truncation bounds for the Taylor series (21) and its parameter derivatives can be obtained using the Cauchy integral formula: for $n \geq 1$,

$$c_{j,n}(u) = -\left(\frac{d}{du}\right)^{n-1} \frac{u^{a-1}\log^j(u)e^{-u}}{j!n!} = -\frac{1}{2\pi i}\frac{1}{j!n}\int_\gamma \frac{t^{a-1}\log^j(t)e^{-t}}{(t-u)^n}dt. \tag{25}$$

Lemma 5. *For $a \in \mathbb{C}$, $u > 0$, $j \geq 0$, $n \geq 1$ and $0 < R < u$, the coefficients $c_{j,n}(u)$ (and $c_n(u) = c_{0,n}(u)$) satisfy the bound*

$$|c_{j,n}(u)| \leq \frac{1}{j!n}\frac{1}{R^{n-1}}M_R(u), \quad M_R(u) = \max_{t:|t-u|=R} |t^{a-1}\log^j(t)e^{-t}|. \tag{26}$$

Consequently, for $N \geq 1$ and $|x| < R$, tails of the Taylor series satisfy

$$\left|\sum_{n=N}^{\infty} c_{j,n}(u)x^n\right| \leq \frac{RM_R(u)}{j!N}\frac{C^N}{1-C}, \quad C = \frac{|x|}{R}. \tag{27}$$

To bound $M_R(u)$, it suffices to bound $t^{a-1}\log^j(t)e^{-t}$ on the disk $|t-u| \leq R$ using naive upper bounds for the elementary functions, or using interval arithmetic. A simple algorithm to choose R is to start with $R = (u+|x|)/2$ and iterate $R \leftarrow R/2$ as long as this decreases the bound. The results can be improved slightly with a proper numerical minimization algorithm.

3.2.3. *Overall algorithm.* The final step is to rewrite the series expansions in matrix form and evaluate the matrix products using binary splitting. We sketch the complete algorithm to compute the incomplete gamma function.

We write $x + [\pm\varepsilon]$ to express the use of an enclosure with midpoint x and radius ε to represent an exact quantity.

Algorithm 6 (Bit-burst evaluation of $\Gamma(a,z)$, with $z > 0$)**.**

- Choose initial number of bits $b_1 = 32$ and let $x = z_1 = 2^{b_1}\lfloor z/2^{b_1}\rfloor$.
- If the asymptotic series (20) is accurate enough:
  - Choose N and denote by ε a bound for the remainder term in (20).
  - Compute $P = U_{N-1}\cdots U_1U_0$ using binary splitting, where

$$U_n = \begin{pmatrix} \frac{a-n-1}{x} & 0 \\ 1 & 1 \end{pmatrix}.$$

  - Now $P_{2,1} = \sum_{n=0}^{N-1} \frac{(1-a)_n}{(-x)^n}$. Compute $y_1 = x^{a-1}e^{-x}(P_{2,1} + [\pm\varepsilon])$, which equals $\Gamma(a,z_1)$.
- Otherwise:
  - Choose N and denote by ε a bound for the remainder term in (17).

  - Compute $P = U_{N-1} \cdots U_1 U_0$ using binary splitting, where
$$U_n = \begin{pmatrix} \frac{x}{a+n+1} & 0 \\ 1 & 1 \end{pmatrix}.$$
  - Now $P_{2,1} = \sum_{n=0}^{N-1} \frac{x^n}{(a+1)_n}$. Compute $y_1 = \Gamma(a) - x^a e^{-x}(P_{2,1} + [\pm\varepsilon])/a$, which equals $\Gamma(a, z_1)$.
  - (At a pole of the gamma function, perform the formal limit computation in the above steps.)
- For $k = 2, 3, \ldots$ with $b_k = 2b_1$, perform the following:
  - Let $z_k = 2^{b_k} \lfloor z / 2^{b_k} \rfloor$. If this approximates z to within the target precision, set $z_k = z$ instead and make this the last iteration.
  - Compute $x = z_k - z_{k-1}$.
  - Choose N and denote by ε a bound for the remainder term in (21).
  - Compute $P = U_{N-1} \cdots U_1 U_0$ using binary splitting, where
$$U_n = \begin{pmatrix} 0 & x & 0 \\ \frac{xn}{Q} & \frac{x(n+1)(n+1+z_{k-1}-a)}{Q} & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad Q = -z_{k-1}(n+1)(n+2).$$
  - Compute $y'_{k-1} = -z_{k-1}^{a-1} e^{-z_{k-1}}$.
  - Compute $y_k = P_{3,1} y_{k-1} + P_{3,2} y'_{k-1} + [\pm\varepsilon]$, which equals $\Gamma(a, z_k)$.
- Return y_k, which equals $\Gamma(a, z)$.

As noted previously, derivatives up to order j with respect to a can be computed using the same algorithm by substituting $a \to a + X$ and working in $\mathbb{C}[[X]]/\langle X^{j+1} \rangle$.

Let us elaborate on the technical details in the proof of Theorem 1. When computing the values $\Gamma((s+\delta)/2, z)$ and $\Gamma((1-s+\delta)/2, z)$ used in the approximate functional equation, the recurrence matrices in Algorithm 6 will have entries in the number field $\mathbb{Q}(s)$, or in the power series ring $R = \mathbb{Q}(s)[[X]]/\langle X^{j+1} \rangle$ if we compute derivatives. When both j and the degree of s are fixed, this ring is a finite-dimensional vector space over $\mathbb{Q}$, and if the minimal polynomial of s has height h, the product or sum of N entries in R with b-bit coefficients will have coefficients with $O(N(b + \log h))$-bit numerators and denominators.

Summing over all bit sizes in the recursion trees for the binary splitting and the consecutive bit-burst steps and using the bound $b^{1+o(1)}$ for the bit complexity of arithmetic on b-bit rational numbers, we obtain the $p^{1+o(1)}$ complexity bound for each call to Algorithm 6.

3.2.4. *Implementarion remarks.* In practice, we should clear denominators so that the matrices have integral entries in the binary splitting products. The products should then be computed using with truncation (rounding) to reduce memory usage and improve performance [Mez12]. Further constant-factor savings are possible by eliminating various redundant computations in the binary splitting process.

The gamma function $\Gamma(a)$ with algebraic a can be computed in quasilinear time by evaluating $\Gamma(a, N)$ with a sufficiently large N using binary splitting [Bre76b, Joh21]. In any case, this only needs to be done once: the same $\Gamma(a)$ value can be recycled for all evaluations of $\Gamma(a, z)$.

When using (17), there can be significant cancellation between the gamma function and the series. This does not affect the *absolute* error when s is small, but

TABLE 1. Time in seconds to compute values of L-functions at fixed simple rational points, using Euler-Maclaurin summation (EM) and the approximate functional equation (AFE).

Digits	Number	EM	AFE	Number	EM	AFE
10^3	$\zeta(1/2)$	0.0076	0.037	$L(1/2, \chi_{23.19})$	0.15	0.18
$\lfloor 10^{3.5} \rfloor$		0.19	0.29		2.3	1.5
10^4		2.7	2.7		38	14
$\lfloor 10^{4.5} \rfloor$		52	27		621	131
10^5		887	262			1282
$\lfloor 10^{5.5} \rfloor$			2175			
10^6			17004			
10^3	$\zeta(4/3)$	0.014	0.083	$L(4/3, \chi_{23.19})$	1.5	0.38
$\lfloor 10^{3.5} \rfloor$		0.34	0.68		39	3.5
10^4		6.0	7.2		795	30
$\lfloor 10^{4.5} \rfloor$		100	66			295
10^5		1808	618			2821
$\lfloor 10^{5.5} \rfloor$			5242			

when s is large, we need to increase the working precision to compensate. The precision increases with z up to the point where we can switch to the asymptotic series (20).

In the pseudocode for Algorithm 6, we evaluate the first derivative $y'(z) = -z^{a-1}e^{-z}$ in each Taylor step. There are several ways to do this: we can compute the elementary functions from scratch, we can perform bit-burst analytic continuation of the function $y'(z)$, or we can perform bit-burst evaluation of the factors z^{a-1} and e^{-z} separately using the standard binomial and exponential Taylor series. Which method performs better may depend on several factors, but either approach achieves quasilinear complexity.

In the approximate functional equation, we need to evaluate $\Gamma(a, z)$ for successive values $z = Cn^2$. It is tempting to reuse the computed $\Gamma(a, z)$ values, starting the bit-burst evaluation at $z_{\text{initial}} = C(n-1)^2$. However, this appears to be a net slowdown, the main reason being that the recurrence matrices are much simpler for the hypergeometric series at the origin than for the expansions at generic points.

4. Implementation results

We have implemented the algorithm for $L(s, \chi)$ with $s \in \mathbb{Q}$ in Arb [Joh17]. We leave an implementation for $s \in \overline{\mathbb{Q}}$ and for $L^{(j)}(s, \chi)$ for future work.

4.1. **Fixed rational points.**

Table 1 illustrates the precision-dependent scaling of the implementation of the approximate functional equation (AFE) when s is a fixed simple fraction. We also show timings for the Euler-Maclaurin implementation of Dirichlet L-functions in Arb (EM).[3]

We observe that the AFE is competitive from about 10^4 digits for computing the Riemann zeta function. The advantage is greater for L-functions with larger modulus q due to the $O(q^{1/2})$ scaling.

At high enough precision, the subquadratic asymptotic complexity of the AFE is evident since the measured time increases by (barely) less than a factor 10 when

[3]The benchmarks were run on a 1.90 GHz Intel i5-4300U CPU.

TABLE 2. Time in seconds to compute the Bernoulli number B_n and Euler number E_n using Harvey's multimodular algorithm (MM), the Euler product (EP), and the approximate functional equation (AFE). Timings marked * were estimated based on the time to evaluate a sparse subsequence (1/10 or 1/100) of the terms, giving an accurate estimate of the time for the full computation without performing it. We indicate the number of digits in the numerator of B_n and in E_n.

Number	n	Digits	MM	EP	AFE
B_n	10^3	1779	0.0066	0.00010	0.067
	$\lfloor 10^{3.5} \rfloor$	7180	0.025	0.0011	0.83
	10^4	27691	0.10	0.012	11
	$\lfloor 10^{4.5} \rfloor$	103330	0.47	0.18	142
	10^5	376772	2.7	1.9	1707
	$\lceil 10^{5.5} \rceil$	1349518	22	21	16578
	10^6	4767554	222	224	159945*
	$\lceil 10^{6.5} \rceil$	16657389	2329	2567	1587800*
E_n	10^3	2372		0.00026	0.19
	$\lfloor 10^{3.5} \rfloor$	9076		0.0026	2.0
	10^4	33699		0.033	24
	$\lfloor 10^{4.5} \rfloor$	122367		0.49	293
	10^5	436962		5.9	2874
	$\lceil 10^{5.5} \rceil$	1539903		68	
	10^6	5369590		726	

the precision is multiplied by $10^{1/2}$. Asymptotically for a $p^{3/2+o(1)}$ complexity algorithm, the time should only increase by a factor $10^{3/4} \approx 5.6$, but the tested precisions are small enough for the hidden logarithmic factors in the complexity bounds to influence the running time. The timings for the $p^{2+o(1)}$ EM algorithm also increase by factors somewhat larger than 10 for the same reason.

There is roughly a factor two slowdown with both algorithms going from $\zeta(1/2)$ to $\zeta(4/3)$, for different reasons: the AFE is inherently twice as fast for $\zeta(1/2)$ since only one of the two series has to be computed; the slowdown with EM is an implementation artifact (the Arb code is not optimized for rational powers).

The million-digit computation of $\zeta(1/2)$, which takes less than five hours on a single core and requires negligible memory, appears to be a precision record for a zeta constant not at an integer.[4]

4.2. **Computation of Bernoulli numbers.** Table 2 compares three algorithms to compute B_n as an exact fraction:

- MM: Harvey's implementation of his $n^{2+o(1)}$ multimodular algorithm [Har10] (available in the `bernmm` module in SageMath).
- EP: the classical $n^{2+o(1)}$ zeta function algorithm using the Euler product implemented in Arb.
- AFE: the approximate functional equation implemented in Arb.

[4]In 2013, the author computed the first nontrivial zero $\frac{1}{2} + 14.134\ldots i$ of $\zeta(s)$ to 303,000 digits using Euler-Maclaurin summation. This took 20 hours and used 62 GB of memory, the high memory usage being the main obstacle to reaching higher precision [Joh14b] (this is an implementation problem that can be avoided).

The last two implementations also support computing Euler numbers.

The MM and EP algorithms both scale superquadratically with n, the multimodular algorithm having a slight edge for n larger than 10^4. This confirms the observations in [Har10, Table 1]. The AFE appears to scale weakly subquadratically, but for reasonably sized n, it is roughly 10^3 times slower than the Euler product. To explain this gap we need to consider the logarithmic and constant factor overheads that we have neglected in the complexity analysis so far.

An analysis with Stirling's formula shows that the cutoff in the Euler product for computing $\zeta(n)$ to $\log_2 |B_n|$ bits of accuracy is $N \approx n/(2\pi e)$, and there are about $N/\log N$ primes up to this cutoff. A similar analysis for the AFE gives the cutoff $N \approx n^{1/2}(2\pi)^{-1/2}\log(n/(2e))$, where all terms are needed, and there are two such series to compute. Considering these facts alone, the AFE thus saves at most a factor $n^{1/2}/(2e\sqrt{2\pi}\log^2 n) \approx n^{1/2}/(13.6\log^2 n)$ asymptotically, which means we need $n \approx 10^7$ to break even *assuming that the terms have unit cost.*

The last assumption is obviously false: series of matrix products are roughly $O(\log^2 n)$ slower than integer powers. An asymptotic speedup of $n^{1/2}/(C\log^4 n)$, $C > 10^1$, is consistent with the observed three-orders-of-magnitude slowdown for $n \approx 10^6$ and suggests that we may need n larger than 10^{15} for the AFE to win, though there is too much uncertainty to extrapolate reliably.

An interesting question is whether Harvey's subquadratic algorithm for Bernoulli numbers [Har14] can perform better, but unfortunately no implementation exists.

5. Evaluation for non-algebraic s

If s is not algebraic and instead must be represented by a p-bit floating-point approximation, then the bit-burst algorithm does not offer any improvement over naive series evaluation since the recurrence matrices will not have small entries, and we only obtain a $p^{5/2+o(1)}$ algorithm to compute $L(s,\chi)$ via (2). However, there at least four independent ways to reduce the complexity to $p^{2+o(1)}$:

(1) We evaluate each $\Gamma(a,z)$ via (17) (optionally together with (20)) using a baby-step giant technique, exploiting the hypergeometric structure of the terms: if the truncated series is represented as a matrix product $\prod_{n=0}^{N-1} U_n$ of length $N = m^2$, we expand $\prod_{n=0}^{m-1} U_{x+n}$ as a matrix of rational functions in x and evaluate at m points using fast multipoint evaluation [Bor87]. This achieves $p^{3/2+o(1)}$ complexity for each incomplete gamma function. This is the approach described in [BBC00].

(2) We expand a truncation of the series (17) (optionally together with (20)) as a polynomial in z of degree $p^{1+o(1)}$ and evaluate it simultaneously at the requisite $p^{1/2+o(1)}$ values of z using fast multipoint evaluation.

(3) Instead of using the series in incomplete gamma function, we use the integral form (3) and its analog for Dirichlet L-functions. Using a standard numerical integration method with geometric rate of convergence for analytic functions, for example Gaussian, Clenshaw-Curtis or double exponential quadrature, we need $p^{1+o(1)}$ evaluations of the integrand. Evaluating the theta function in the integrand using the q-series costs $p^{1/2+o(1)}$ multiplications, resulting in an $p^{5/2+o(1)}$ algorithm for $L(s,\chi)$. This is the method used by Keiper [Kei92]. However, we can compute the theta function in quasilinear time using arithmetic-geometric mean iteration instead [Lab18], and this achieves $p^{2+o(1)}$ complexity for $L(s,\chi)$.

(4) As above, but we expand the truncated q-series for the theta function as a polynomial and evaluate it at all the integration nodes using fast multipoint evaluation.

The techniques for fast evaluation of theta functions used in methods (3) and (4) have previously been used in the context of computing class polynomials via numerical approximations of the roots [Eng09].

We also mention a version of method (2) that is asymptotically slower but may be superior at realistic levels of precision. We can expand a truncation of the series (17) (and optionally (20)) for $\Gamma(a, \pi n^2 \alpha/q)$ as a polynomial in n^2. This reduces the computation to multipoint evaluation at the small integers $n = 1, 2, \ldots$, which may be performed using repeated applications of Horner's rule instead of fast multipoint evaluation. This results in a $p^{5/2+o(1)}$ algorithm but with very little overhead since the Horner evaluations only involve additions and p-by-1-word multiplications which are orders of magnitude cheaper than full p-by-p multiplications.

Yet another option is the Booker-Molin method, which employs a Fourier series that can be precomputed for efficient evaluation at many values of s [BC21, §9.4].

It is not clear *a priori* which of the above methods will perform better, so further implementation studies are needed.

Did Borwein and Borwein [BB88] have one of the methods above in mind for non-algebraic s, or did they have an entirely different algorithm? It is likely that they considered (1) or (2) since their paper discusses the same multipoint evaluation techniques for other functions, although their wording is more suggestive of an algorithm along the lines of (3) or (4). Either way, there seems to be no obvious way to obtain a subquadratic algorithm for $\zeta(s)$ or $L(s, \chi)$ for non-algebraic s, and this remains an open problem along with the problem of finding a quasilinear algorithm for $s \in \overline{\mathbb{Q}} \setminus \mathbb{Z}$.

Acknowledgements

The author was supported in part by the ANR grant ANR-20-CE48-0014-02 NuSCAP.

References

[AB13] George E. Andrews and Bruce C. Berndt. *Ramanujan's Lost Notebook, Part IV*. Springer New York, 2013.

[BB88] Jonathan M. Borwein and Peter B. Borwein. On the complexity of familiar functions and numbers. *SIAM Review*, 30(4):589–601, December 1988.

[BB01] David H. Bailey and Jonathan M. Borwein. Experimental mathematics: Recent developments and future outlook. pages 51–66. Springer Berlin Heidelberg, 2001.

[BB15] David H Bailey and Jonathan M Borwein. Crandall's computation of the incomplete gamma function and the Hurwitz zeta function, with applications to Dirichlet L-series. *Applied Mathematics and Computation*, 268:462–477, 2015.

[BBC00] J. M. Borwein, D. M. Bradley, and R. E. Crandall. Computational strategies for the Riemann zeta function. *Journal of Computational and Applied Mathematics*, 121:247–296, 2000.

[BC21] Karim Belabas and Henri Cohen. *Numerical Algorithms for Number Theory: Using Pari/GP*, volume 254. American Mathematical Society, 2021.

[BM80] R. P. Brent and E. M. McMillan. Some new algorithms for high-precision computation of Euler's constant. *Mathematics of Computation*, 34(149):305–312, 1980.

[Boo06] Andrew R Booker. Artin's conjecture, Turing's method, and the Riemann hypothesis. *Experimental Mathematics*, 15(4):385–407, 2006.

[Bor87] P. B. Borwein. Reduced complexity evaluation of hypergeometric functions. *Journal of Approximation Theory*, 50(3):193–199, July 1987.

[Bre76a] R. P. Brent. The complexity of multiple-precision arithmetic. *The Complexity of Computational Problem Solving*, pages 126–165, 1976.

[Bre76b] Richard P. Brent. Fast multiple-precision evaluation of elementary functions. *Journal of the ACM*, 23(2):242–251, April 1976.

[CC90] D. V. Chudnovsky and G. V. Chudnovsky. Computer algebra in the service of mathematical physics and number theory. *Computers in mathematics*, 125:109, 1990.

[Cof14] Mark W Coffey. Series representations for the Stieltjes constants. *Rocky Mountain Journal of Mathematics*, 44(2):443–477, 2014.

[Coh19] Henri Cohen. Computational number theory in relation with L-functions. In *Notes from the International Autumn School on Computational Number Theory*, pages 171–266. Springer International Publishing, 2019.

[Cra12] R Crandall. Unified algorithms for polylogarithm, L-series, and zeta variants. *Algorithmic Reflections: Selected Works. PSIpress*, 2012.

[Dok04] Tim Dokchitser. Computing special values of motivic L-functions. *Experimental Mathematics*, 13(2):137–149, 2004.

[Eng09] Andreas Enge. The complexity of class polynomial computation via floating point approximations. *Mathematics of Computation*, 78(266):1089–1107, 2009.

[Fin03] Steven R Finch. *Mathematical constants*. Cambridge university press, 2003.

[FV96] Philippe Flajolet and Ilan Vardi. Zeta function expansions of classical constants. 1996.

[Har10] David Harvey. A multimodular algorithm for computing Bernoulli numbers. *Mathematics of Computation*, 79(272):2361–2361, 2010.

[Har14] David Harvey. A subquadratic algorithm for computing the n-th Bernoulli number. *Mathematics of Computation*, 83(289):2471–2477, April 2014.

[HvdH21] David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193(2):563, 2021.

[JB18] F. Johansson and I. V. Blagouchine. Computing Stieltjes constants using complex integration, 2018. https://arxiv.org/abs/1804.01679.

[Joh14a] F. Johansson. *Fast and rigorous computation of special functions to high precision*. PhD thesis, RISC, Johannes Kepler University, Linz, 2014.

[Joh14b] Fredrik Johansson. Rigorous high-precision computation of the Hurwitz zeta function and its derivatives. 69(2):253–270, July 2014.

[Joh17] Fredrik Johansson. Arb: Efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Transactions on Computers*, 66(8):1281–1292, August 2017.

[Joh19] Fredrik Johansson. Computing hypergeometric functions rigorously. *ACM Transactions on Mathematical Software*, 45(3):1–26, August 2019.

[Joh21] Fredrik Johansson. Arbitrary-precision computation of the gamma function. 2021.

[Kar98] E. A. Karatsuba. Fast evaluation of the Hurwitz zeta function and Dirichlet L-series. *Problems of Information Transmission*, 34(4):62–75, 1998.

[Kei92] J. B. Keiper. Power series expansions of riemann's ξ function. *Mathematics of Computation*, 58(198):765–773, 1992.

[Lab18] Hugo Labrande. Computing Jacobi's theta in quasi-linear time. *Mathematics of Computation*, 87(311):1479–1508, 2018.

[LT72] J. J. Y. Liang and J. Todd. The Stieltjes constants. *Journal of Research of the National Bureau of Standards*, 76:161–178, 1972.

[Mez11] M. Mezzarobba. *Autour de l'évaluation numérique des fonctions D-finies*. Thèse de doctorat, Ecole polytechnique, November 2011.

[Mez12] Marc Mezzarobba. A note on the space complexity of fast D-finite function evaluation. In *International Workshop on Computer Algebra in Scientific Computing*, pages 212–223. Springer, 2012.

[Mol10] Pascal Molin. *Intégration numérique et calculs de fonctions L*. PhD thesis, Université Sciences et Technologies-Bordeaux I, 2010.

[Nat13] National Institute of Standards and Technology. Digital Library of Mathematical Functions. http://dlmf.nist.gov/, 2013.

[Pla11] David J Platt. *Computing degree 1 L-functions rigorously*. PhD thesis, University of Bristol, 2011.

[Rie59] Bernhard Riemann. Ueber die Anzahl der Primzahlen unter einer gegebenen Grosse. *Ges. Math. Werke und Wissenschaftlicher Nachlaß*, 2:145–155, 1859.

[Rub98] Michael Oded Rubinstein. *Evidence for a spectral interpretation of the zeros of L-functions*. Princeton University, 1998.

[SG03] Pascal Sebah and Xavier Gourdon. The Apery's constant: $\zeta(3)$. http://numbers.computation.free.fr/Constants/Zeta3/zeta3.html, 2003.

[vdH99] J. van der Hoeven. Fast evaluation of holonomic functions. *Theoretical Computer Science*, 210:199–215, 1999.

[vdH01] J. van der Hoeven. Fast evaluation of holonomic functions near and in regular singularities. *Journal of Symbolic Computation*, 31(6):717–743, 2001.

[vdH09] J. van der Hoeven. Ball arithmetic. Technical report, HAL, 2009. http://hal.archives-ouvertes.fr/hal-00432152/fr/.

[Yee21] Alexander J. Yee. y-cruncher - a multi-threaded pi-program. http://www.numberworld.org/y-cruncher/, 2021.

Inria Bordeaux, 33400 Talence, France
Email address: fredrik.johansson@gmail.com