MATHEMATICS OF COMPUTATION
Volume 80, Number 275, July 2011, Pages 1785–1796
S 0025-5718(2011)02452-X
Article electronically published on January 25, 2011

AN AMORTIZED-COMPLEXITY METHOD TO COMPUTE THE RIEMANN ZETA FUNCTION

GHAITH A. HIARY

ABSTRACT. A practical method to compute the Riemann zeta function is presented. The method can compute $\zeta(1/2+it)$ at any $\lfloor T^{1/4} \rfloor$ points in $[T, T+T^{1/4}]$ using an *average* time of $T^{1/4+o(1)}$ per point. This is the same complexity as the Odlyzko-Schönhage algorithm over that interval. Although the method far from competes with the Odlyzko-Schönhage algorithm over intervals much longer than $T^{1/4}$, it still has the advantages of being elementary, simple to implement, it does not use the fast Fourier transform or require large amounts of storage space, and its error terms are easy to control. The method has been implemented, and results of timing experiments agree with its theoretical amortized complexity of $T^{1/4+o(1)}$.

1. INTRODUCTION

The Riemann zeta function $\zeta(s)$ can be calculated using the Riemann-Siegel formula. A frequently stated version of that formula on the critical line is: Let

$$Z(t) = e^{i\theta(t)}\zeta(1/2+it)\,, \qquad e^{i\theta(t)} = \left(\frac{\Gamma(1/4+it/2)}{\Gamma(1/4-it/2)}\right)^{1/2} \pi^{-it/2}\,. \tag{1.1}$$

The rotation factor $e^{i\theta(t)}$ is chosen so that $Z(t)$ is real. Let $a := \sqrt{t/(2\pi)}$, $n_1 := \lfloor a \rfloor$ be the integer part of a, and $\rho := \{a\} = a - \lfloor a \rfloor$ the fractional part of a. Then for $t > 2\pi$,

$$\begin{aligned} Z(t) = \Re\left(2e^{-i\theta(t)}\sum_{n=1}^{n_1}\frac{e^{it\log n}}{\sqrt{n}}\right) + \frac{(-1)^{n_1+1}}{a^{1/2}}\Phi(\rho) \\ + \frac{(-1)^{n_1+2}}{96\pi^2 a^{3/2}}\Phi^{(3)}(\rho) + R(t)\,, \end{aligned} \tag{1.2}$$

where

$$\Phi(x) := \frac{\cos 2\pi(x^2-x-1/16)}{\cos 2\pi x}\,, \tag{1.3}$$

and $\Phi^{(3)}(x)$ is the third derivative of $\Phi(x)$ with respect to x. Gabcke [Ga] showed

$$|R(t)| \le .053t^{-5/4}, \quad \text{for } t \ge 200\,, \tag{1.4}$$

Received by the editor February 11, 2010 and, in revised form, April 28, 2010.
2000 *Mathematics Subject Classification.* Primary 11M06, 11Y16; Secondary 68Q25.
Key words and phrases. Riemann zeta function, algorithms.
This material is based upon work supported by the National Science Foundation under agreements No. DMS-0757627 (FRG grant) and No. DMS-0635607.

which is sufficient for most applications. Odlyzko and Schönhage [OS] showed how to compute the rotation factor $e^{-i\theta(t)}$, and the correction terms $\Phi(x)$ and $\Phi^{(3)}(x)$, to within $\pm t^{-\kappa-10}$, with $\kappa > 0$ fixed, using $t^{o_\kappa(1)}$ operations on numbers of $O_\kappa(\log t)$ bits. (Note throughout, asymptotic constants are taken as $t \to \infty$, and the notations $O_\kappa(.)$ and $o_\kappa(.)$ mean asymptotic constants depend on the parameter κ only.) Therefore, to calculate the rotated zeta function $Z(t)$, the bulk of the computational effort is in computing the sum

$$F(t) := \sum_{n=1}^{n_1} \frac{e^{it\log n}}{\sqrt{n}}\,, \qquad n_1 = \lfloor \sqrt{t/(2\pi)} \rfloor\,. \tag{1.5}$$

By taking more correction terms in formula (1.2), one can arrange for the remainder term $R(t)$ to be bounded by $O(t^{-\kappa})$ for any fixed $\kappa > 0$. Odlyzko and Schönhage [OS] showed the additional correction terms can also be computed to within $\pm t^{-\kappa}$ in $t^{o_\kappa(1)}$ operations on numbers of $O_\kappa(\log t)$ bits.

Frequently, one is interested in numerically evaluating $Z(t)$ at N points in an interval of the form $t \in [T, T + T^\alpha]$, where $\alpha \in [0, 1/2]$ say, and N is large. This is the case, for example, when one attempts to locate real zeros of $Z(t)$ to within $\pm T^{-\kappa}$, or study moments of the zeta function. A straightforward application of the Riemann-Siegel formula can do so in $N\,T^{1/2+o_\kappa(1)}$ operations.

The purpose of this paper is to improve the running time in the N-aspect. So, although the proposed method still consumes $t^{1/2+o_\kappa(1)}$ time to evaluate $Z(t)$ to within $\pm t^{-\kappa}$ at a *single* point, it achieves substantially lower running times if one is interested in computing zeta at *many* points. This type of idea is not new: in the context of the Riemann zeta function, it dates back to the algorithm of Odlyzko and Schönhage [OS].

The main feature of the proposed method is that it completely avoids the two essential components of the Odlyzko-Schönhage algorithm, which are the fast Fourier transform, and a sophisticated rational function evaluation algorithm. Instead, the method relies on a straightforward subdivision of the main sum in the Riemann-Siegel formula, a band-limited interpolation technique (see Appendix), and a direct evaluation to obtain the precomputation data. Therefore, its implementation is relatively straightforward, with friendly asymptotic constants, and its error terms are easy to bound. Also, it does not require large amounts of storage space for its precomputation data. The method far from competes with the Odlyzko-Schönhage algorithm in general; but in many situations, it achieves a similar complexity.

The basic idea of the method is that computing $Z(t)$ for many different, but neighboring, values of t involves many common steps. The method takes advantage of this to achieve lower running times.

Before discussing the method any further, we make a few remarks. By *computing* zeta we mean to numerically evaluate $Z(t)$ with "polynomial accuracy", that is, with an absolute error bounded by $t^{-\kappa}$, for any fixed $\kappa > 0$. We measure the computational complexity of our method by the number of arithmetic operations required: additions, multiplications, divisions, complex exponential, and logarithm (involving numbers of $O_\kappa(\log t)$ bits). That in turn can be routinely bounded by the number of bit operations. Last, as in the Odlyzko-Schönhage algorithm, the proposed method will generalize easily to any Dirichlet series:

$$\sum_n \frac{a_n}{n^s}\,, \qquad s = \sigma + it \ \text{ with } \ \sigma \in [0,1] \ \text{ say}\,, \tag{1.6}$$

assuming the coefficients a_n are known, or can be computed quickly. For definiteness, in the remainder of the paper, we specialize to the rotated zeta function on the critical line $Z(t)$. Our main result is the following.

Theorem 1.1. *Given any fixed numbers $\alpha \in [0, 1/2]$ and $\kappa > 0$, there exists an algorithm that for every $T > 100$ will perform $T^{1/2+o_\kappa(1)}$ operations on numbers of $O_\kappa(\log T)$ bits using $T^{\alpha+o_\kappa(1)}$ bits of storage, after which the algorithm will be capable of computing $Z(t)$ at any $t \in [T, T + T^\alpha]$ to within $\pm T^{-\kappa}$ in $T^{\alpha+o_\kappa(1)}$ operations.*

It is useful to compare the algorithm of Theorem 1.1 with the Odlyzko-Schönhage algorithm (the statement below is equivalent to Theorem 5.1 in [OS] specialized to $Z(t)$):

The Odlyzko-Schönhage algorithm. *Given any $a \in [0, 1/2]$, and any constants ϵ and κ, there is an effectively computable constant $B = B(\epsilon, \kappa, a)$, and an algorithm that for every $T > 0$ will perform $\leq BT^{1/2+\epsilon}$ operations on numbers of $\leq B \log T$ bits using $\leq BT^{a+\epsilon}$ bits of storage and will then be capable of computing any value of $Z(t)$ for $T \leq t \leq T + T^a$ to within $\pm T^{-\kappa}$ in $\leq BT^\epsilon$ operations using the precomputed values.*

As mentioned earlier, the two central ingredients of the Odlyzko-Schönhage algorithm are the fast Fourier transform, and a rational function evaluation algorithm. Our method completely avoids these central ingredients. Instead, it relies on a straightforward subdivision of the main sum in the Riemann-Siegel formula, a band-limited interpolation technique (see Appendix), and a direct evaluation for the precomputation data. So, it is significantly simpler.

We remark that the Odlyzko-Schönhage algorithm has been implemented at least twice, by Odlyzko [O], and by Gourdon [G]. Gourdon's implementation replaces the Odlyzko-Schönhage rational function algorithm by the Greengard-Rokhlin algorithm. This was suggested by Odlyzko as a possible improvement in [O].

We carry out a comparison between the Odlyzko-Schönhage algorithm and the algorithm of Theorem 1.1 in the following context. Suppose we wish to evaluate $Z(t)$ to within $\pm T^{-\kappa}$ at about $\lfloor T^\alpha \rfloor$ points in the interval $[T, T + T^\alpha]$. This is often the case in applications; for example, when one attempts to locate, to within $\pm T^{-\kappa}$, the ordinates of non-trivial zeros of zeta on the critical line (notice these are the real zeros of $Z(t)$), one expects to require about $T^{o_\kappa(1)}$ evaluations of $Z(t)$ per zero. Since there are $T^{\alpha+o(1)}$ zeros of $Z(t)$ in the interval $[T, T + T^\alpha]$, then about $T^{\alpha+o_\kappa(1)}$ evaluations of $Z(t)$ are needed in total. So, consider Table 1. It compares the running times of the algorithm of Theorem 1.1 and the Odlyzko-Schönhage algorithm in such a situation.

TABLE 1

Algorithm	Precomputation	Storage	Single Eval.	Average
Theorem 1.1	$T^{1/2}$	T^α	T^α	$T^{1/2-\alpha} + T^\alpha$
Odlyzko-Schönhage	$T^{1/2+\epsilon}$	$T^{\alpha+\epsilon}$	T^ϵ	$T^{1/2-\alpha+\epsilon} + T^\epsilon$

In Table 1, the column "Single Eval." refers to the cost of a single evaluation of $Z(t)$ after the precomputation has been carried out. The column "Average" refers

to the average cost of evaluating $Z(t)$ at all $\lfloor T^\alpha \rfloor$ points (in particular, the column "Average" takes the precomputation cost into account). To avoid notational clutter, the complexities listed in the table ignore little-o terms of the form $T^{o(1)}$ (In [OS], it is stated the T^ϵ term can be replaced by a fixed power of $\log T$; nevertheless, it is included in the table to be consistent with the formal statement of the Odlyzko-Schönhage algorithm).

In view of the last column of the table, we see if $\alpha \le 1/4$, the algorithm of Theorem 1.1 achieves the same amortized complexity as the Odlyzko-Schönhage algorithm, but when $\alpha > 1/4$ it does not. This suggests one should restrict $\alpha \in [0, 1/4]$. More generally, we have the following corollary to Theorem 1.1.

Corollary 1.2. *Given any fixed numbers $a \in [0, 1/2]$ and $\kappa > 0$, there exists an algorithm that for every $T > 100$, will be capable of computing $Z(t)$ to within $\pm T^{-\kappa}$ at any $\lfloor T^a \rfloor$ points in $[T, T + T^a]$ using an average of*

$$\begin{cases} T^{1/2-a+o_\kappa(1)} & \text{operations per point if } 0 \le a \le 1/4, \\ T^{1/4+o_\kappa(1)} & \text{operations per point if } 1/4 < a \le 1/2, \end{cases}$$

where the operations are performed on numbers of $O_\kappa(\log T)$ bits. The storage space requirement for the algorithm is

$$\begin{cases} T^{a+o_\kappa(1)} & \text{bits if } 0 \le a \le 1/4, \\ T^{1/4+o_\kappa(1)} & \text{bits if } 1/4 < a \le 1/2. \end{cases}$$

By comparison, to accomplish the same task as in the statement of the corollary, the Odlyzko-Schönhage algorithm requires $T^{1/2+\epsilon-a+o_{\epsilon,\kappa,a}(1)}$ operations per point on average, and it requires a storage space of $T^{a+\epsilon+o_{\epsilon,\kappa,a}(1)}$ bits (which is the same as Corollary 1.2 when $a \in [0, 1/4]$).

The statement of Corollary 1.2 follows from a straightforward optimization procedure. Specifically, for any fixed numbers $\delta \ge 0$, $\alpha \in [0, 1/2]$, and $\kappa > 0$, such that $\delta + \alpha \in [0, 1/2]$, consider the successive intervals

$$[T + jT^\alpha, T + (j+1)T^\alpha], \qquad j = 0, 1, \dots, \lceil T^\delta \rceil. \tag{1.7}$$

Then, applying the algorithm of Theorem 1.1 a total of $\lceil T^\delta \rceil + 1$ times to these intervals, we deduce $Z(t)$ can be computed to within $\pm T^{-\kappa}$ at any single point in

$$[T, T + T^{\alpha+\delta}], \qquad \alpha \in [0, 1/2], \qquad \delta \ge 0, \qquad \alpha + \delta \in [0, 1/2], \tag{1.8}$$

using $T^{\alpha+o_\kappa(1)}$ operations, on numbers of $O_\kappa(\log T)$ bits, provided a precomputation costing $T^{1/2+\delta+o_\kappa(1)}$ operations is performed. Notice the storage space requirement for the precomputation data can always be kept at $T^{\alpha+o_\kappa(1)}$ bits. This is because the method will deal with one subinterval $[T + jT^\alpha, T + (j+1)T^\alpha]$ at a time, so the precomputation data for that subinterval can be discarded when the method is done there.

Now, as in the statement of the corollary, let $a \in [0, 1/2]$, and suppose we wish to compute $Z(t)$ to within $\pm T^{-\kappa}$ at $\lfloor T^a \rfloor$ points in the interval $[T, T+T^a]$. Optimizing α and δ to the above situation, we solve

$$1/2 + \delta = a + \alpha, \qquad \text{and} \qquad \alpha + \delta = a, \tag{1.9}$$

which has the solution $\alpha = 1/4$ and $\delta = a - 1/4$. If $a < 1/4$, then δ becomes negative. So in this case, we choose $\alpha = a$ and $\delta = 0$. For example, when $a \in [0, 1/4)$, the method requires $T^{1/2+\delta+o_\kappa(1)} = T^{1/2+o_\kappa(1)}$ operations on numbers of $O_\kappa(\log T)$ bits to compute $Z(t)$ to within $\pm T^{-\kappa}$ at all the $\lfloor T^a \rfloor$ points in the interval $[T, T + T^a]$.

This amounts to an average of $T^{1/2-a+o_\kappa(1)}$ operations per point. The storage space requirements (for the precomputation data) is $T^{\alpha+o_\kappa(1)} = T^{a+o_\kappa(1)}$ bits. The analysis of the case $a \in (1/4, 1/2]$ is analogous.

We compare the efficiency of our proposed method with Gourdon's implementation [G] of the Odlyzko-Schönhage algorithm. To this end, consider that in order to locate zeros near $t = 10^{16}$ to within $\pm 10^{-8}$ say, one expects ≈ 8 evaluations of $Z(t)$ are required per zero (see [O] p. 80, and [G] p. 26). The mean spacing of zeros near 10^{16} is about $2\pi/\log(10^{16}/(2\pi)) \approx 0.18$. Therefore, one expects to evaluate $Z(t)$ at points with $1/8 \times 0.18 \approx 0.02$ mean spacing.

In particular, to locate 2×10^9 consecutive zeros near $t = 10^{16}$ one expects to compute $Z(t)$ at $8 \times 2 \times 10^9 = 1.6 \times 10^{10}$ successive points with mean spacing 0.02. Results of timing tests (see last entry in Table 2 and footnote 3 in Section 3), suggest our method will consume about 9 minutes to compute $Z(t)$ at 10^5 such points. By extrapolation, we expect the method to consume $(9/60) \times (1.6 \times 10^5) = 24,000$ hours to compute $Z(t)$ at 1.6×10^{10} such points. So, we extrapolate that the method will require 24,000 hours to locate 2×10^9 zeros of $Z(t)$ near $t = 10^{16}$. By comparison, Gourdon's implementation of the Odlyzko-Schönhage algorithm (see [G], p. 28) consumes 49.5 hours to locate the same number of zeros at that height. Thus, Gourdon's implementation is approximately 485 times faster than our method for that task. In turn, our method is approximately 437 times faster than *lcalc*'s implementation of the direct Riemann-Siegel formula; see Table 2, Section 3.

However, if one is interested in finding a smaller set of zeros, then our method might be more suitable, both in terms of timings and the required programming effort. This is because one does not expect the time requirement of the Odlyzko-Schönhage algorithm to decrease significantly as the size of the set of zeros to be located decreases, whereas the time requirement of our method becomes substantially less. The reason is near $t = 10^{16}$, and with 10^9 zeros to be located, we are working in the region $a \approx 1/2$ in Corollary 1.2. So the running time of our method is roughly linear in the number of zeros to be found. For example, to locate 2×10^8 zeros to within $\pm 10^{-8}$ near $t = 10^{16}$, we expect our method to consume 10 times less than in the previous scenario, or $\approx 2,400$ hours.

We remark that our implementation of the method will be available in the next release of Michael Rubinstein's *lcalc*; see [R1].

2. Proof of Theorem 1.1

In view of formula (1.2), and the remarks made thereafter, in order to compute $Z(t)$ with polynomial accuracy, it suffices to numerically evaluate the main sum

$$F(t) := \sum_{n=1}^{n_1} \frac{e^{it\log n}}{\sqrt{n}}\,, \qquad \text{where } n_1 := \lfloor \sqrt{t/(2\pi)} \rfloor\,, \tag{2.1}$$

with polynomial accuracy. Given $T > 100$, and fixed numbers $\alpha \in [0, 1/2]$, $\kappa > 0$, we wish to evaluate $F(t)$ for many values of $t \in [T, T + T^\alpha]$ to within $\pm T^{-\kappa}$. Let $n_2 := \lfloor \sqrt{T/(2\pi)} \rfloor$. Since $\alpha \in [0, 1/2]$, then n_1, which is the length of the main sum (2.1), is equal to either n_2 or $n_2 + 1$. Without loss of generality, we may assume the length of the main sum is n_2. Let $M := \min\{\lceil T^\alpha \rceil\,, n_2\}$. Note M and n_2 depend only on T. Then, there exist unique integers $M_1, m \geq 0$ (also depending only on

T), with $n_2 = 2^m M + M_1$, and $M_1 < 2^m M$. So the main sum can be written as

$$\begin{aligned}\underbrace{\sum_{1\le n<M} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Initial sum}} + \sum_{M\le n<2M} \frac{e^{it\log n}}{\sqrt{n}} + \cdots \\ + \sum_{2^{m-1}M\le n<2^m M} \frac{e^{it\log n}}{\sqrt{n}} + \underbrace{\sum_{2^m M\le n\le 2^m M+M_1} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Tail sum}}.\end{aligned} \tag{2.2}$$

The "initial sum" in (2.2) can be evaluated directly to within $\pm T^{-\kappa-1}$ in $10M \le 10T^{\alpha}$ operations on numbers of $\lceil (10\kappa+10)\log T\rceil$ bits, say. Thus, we may focus our attention on computing (to within $T^{-\kappa-1}$) the subsums:

$$\sum_{n=2^l M}^{2^{l+1}M-1} \frac{e^{it\log n}}{\sqrt{n}} \qquad l\in\{0,1,\dots,m-1\} \qquad \text{and} \qquad \underbrace{\sum_{n=2^m M}^{2^m M+M_1} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Tail sum}}. \tag{2.3}$$

A direct evaluation of all of the subsums (2.3) requires $T^{1/2+o_k(1)}$ operations. But computing the individual terms in these subsums involves many common steps for many different choices of $t\in[T,T+T^{\alpha}]$. We take advantage of this to obtain substantially lower running times. So consider one of the subsums on the left side of (2.3). Split it into consecutive blocks of length 2^l. This gives,

$$\begin{aligned}\sum_{n=2^l M}^{2^{l+1}M-1} \frac{e^{it\log n}}{\sqrt{n}} = \underbrace{\sum_{n=2^l M}^{2^l M+2^l-1} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{First block}} + \underbrace{\sum_{n=2^l M+2^l}^{2^l M+2(2^l)-1} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Second block}} \\ + \underbrace{\sum_{n=2^l M+2(2^l)}^{2^l M+3(2^l)-1} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Third block}} + \cdots + \underbrace{\sum_{n=2^l M+(M-1)2^l}^{2^l M+M2^l-1} \frac{e^{it\log n}}{\sqrt{n}}}_{M^{th}\text{ block}}.\end{aligned} \tag{2.4}$$

Notice for any $l\in\{0,1,\dots,m-1\}$, we have a total of $2^l M/2^l = M$ blocks. So the total number of blocks for any such l is exactly M. As for the "Tail sum" in (2.3), we use blocks of length 2^m. This gives a similar outcome as in (2.4), except now the total number of blocks is $M' = \lfloor (M_1+1)2^{-m}\rfloor$, which is at most M, and there is possibly a "Remainder block" of length $\le 2^m$,

$$\mathcal{R}_{M,M_1,m}(t) := \underbrace{\sum_{n=2^m M+M'2^m}^{2^m M+M_1} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Remainder block}}, \qquad \text{where } M' := \left\lfloor \frac{M_1+1}{2^m}\right\rfloor \le M. \tag{2.5}$$

Other than the remainder block $\mathcal{R}_{M,M_1,m}(t)$, all our blocks are members of the set

$$\sum_{n=2^l M+r2^l}^{2^l M+(r+1)2^l-1} \frac{e^{it\log n}}{\sqrt{n}}, \qquad l\in\{0,1,\dots,m\},\ r\in\{0,1,\dots,M-1\}. \tag{2.6}$$

(when $l=m$, only the values $r<M'$ are relevant to the algorithm). Let

$$v := v_{l,r} = 2^l M + r2^l, \qquad K := K_l = 2^l. \tag{2.7}$$

Then the v^{th} block (the one starting at $v := v_{l,r}$) can be written in the form

$$\sum_{k=0}^{K-1} \frac{e^{it\log(v+k)}}{\sqrt{v+k}} = e^{it\log v} \sum_{k=0}^{K-1} \frac{e^{it\log(1+k/v)}}{\sqrt{v+k}} =: e^{it\log v} F_{v,K}(t)\,. \tag{2.8}$$

The function $F_{v,K}(t)$ is a *band-limited function*; that is, its spectrum is limited to a finite interval, which in this case is the interval $[0, \log(1+K/v)]$. In more technical terms, a band-limited function can be defined as a function whose Fourier transform is a tempered distribution with compact support. There exists a clever method to compute band-limited functions, which we reproduce in the Appendix with slight modifications (see [O], p. 88, for an in-depth discussion and history of the method). To apply band-limited interpolation, first note by construction, that

$$\frac{K}{v} = \frac{K_l}{v_{l,r}} = \frac{2^l}{2^l M + r2^l} \le \frac{1}{M}\,, \tag{2.9}$$

for all $l \in \{0, 1, \dots, m\}$, and $r \in \{0, 1, \dots, M-1\}$. Therefore, the spectrum of the functions $F_{v,K}(t)$ is always contained in the interval $[0, M^{-1}]$. As for the remainder block $\mathcal{R}_{M,M_1,m}(t)$, let

$$v' := 2^m M + M' 2^m\,, \qquad K' := M_1 - M' 2^m\,, \qquad (\text{note } K' \le 2^m), \tag{2.10}$$

then write

$$\begin{aligned} \mathcal{R}_{M,M_1,m}(t) &:= \sum_{n=v'}^{v'+K'} \frac{e^{it\log n}}{\sqrt{n}} = e^{it\log v'} \sum_{k=0}^{K'} \frac{e^{it\log(1+k/v')}}{\sqrt{v'+k}} \\ &=: e^{it\log v'} F_{v',K'}(t)\,. \end{aligned} \tag{2.11}$$

Then similarly to the bound (2.9), we have

$$\frac{K'}{v'} = \frac{M_1 - M'2^m}{2^m M + M'2^m} \le \frac{2^m}{2^m M + M'2^m} \le \frac{1}{M}\,. \tag{2.12}$$

In particular, the spectrum of $F_{v',K'}(t)$ is also contained in $[0, M^{-1}]$. So, we may apply formulas (4.3) and (4.4) in the Appendix with $G(t) = F_{v,K}(t)$, and $G(t) = F_{v',K'}(t)$, and (in both cases),

$$\tau = M^{-1}\,, \quad \beta = 3\tau\,, \quad \lambda = (\beta+\tau)/2 = 2\tau\,, \quad \epsilon_1 = (\beta-\tau)/2 = \tau\,, \tag{2.13}$$

say. We then appeal to the bounds (4.5) and (4.6) in the Appendix to obtain

$$\begin{aligned} F_{v,K}(t) = \frac{\lambda}{\beta} \sum_{|n\pi/\beta - t| < c/\epsilon_1} F_{v,K}\left(\frac{n\pi}{\beta}\right) \frac{\sin\lambda(n\pi/\beta - t)}{\lambda(n\pi/\beta - t)} h(n\pi/\beta - t) \\ + \mathcal{E}_{v,K}\,. \end{aligned} \tag{2.14}$$

for any $c > 1$, where

$$h(u) := \frac{c}{\sinh(c)} \frac{\sinh\sqrt{c^2 - \epsilon_1^2 u^2}}{\sqrt{c^2 - \epsilon_1^2 u^2}}\,, \qquad |\mathcal{E}_{v,K}| < 6e^{-c} \sum_{k=0}^{K-1} \frac{1}{\sqrt{v+k}}\,. \tag{2.15}$$

By similar calculations, we obtain an analogous formula to (2.14) for $F_{v',K'}(t)$, which corresponds to the remainder block $\mathcal{R}_{M,M_1,m}(t)$. Now, define

$$\tilde{F}_{v,K}(t) = \frac{\lambda}{\beta} \sum_{|n\pi/\beta - t| < c/\epsilon_1} F_{v,K}\left(\frac{n\pi}{\beta}\right) \frac{\sin\lambda(n\pi/\beta - t)}{\lambda(n\pi/\beta - t)} h(n\pi/\beta - t)\,. \tag{2.16}$$

Also define $\tilde{F}_{v',K'}(t)$ in an analogous way to (2.16). Note $\tilde{F}_{v,K}$ (also, $\tilde{F}_{v',K'}(t)$) is a sum of $\le 2c\,\beta/\epsilon_1 \le 6c$ terms. Then put together, we have

$$\begin{aligned}\underbrace{\sum_{1\le n\le\sqrt{T/(2\pi)}} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Main sum}} &= \underbrace{\sum_{1\le n<M} \frac{e^{it\log n}}{\sqrt{n}}}_{\text{Initial sum}} + \sum_{l=0}^{m-1}\sum_{r=0}^{M-1} e^{it\log v_{l,r}}\tilde{F}_{v_{l,r},K_l}(t) \\ &\quad + \sum_{r=0}^{M'-1} e^{it\log v_{m,r}}\tilde{F}_{v_{m,r},K_m}(t) + \tilde{F}_{v',K'}(t) + \mathcal{E}\,,\end{aligned} \tag{2.17}$$

where

$$|\mathcal{E}| \le \sum_{l=0}^{m-1}\sum_{r=0}^{M-1} |\mathcal{E}_{v_{l,r},K_l}| + \sum_{r=0}^{M'-1} |\mathcal{E}_{v_{l,r},K_l}| + |\mathcal{E}_{v',K'}| \le 20e^{-c}\,T^{1/4}. \tag{2.18}$$

We choose $c = (\kappa+2)\log T$, so that $|\mathcal{E}| < T^{-\kappa-1}$. Last, as we will soon explain, the right side of (2.17) can now be evaluated to within $\pm T^{-\kappa}$ using $T^{\alpha+o_\kappa(1)}$ operations provided we precompute the entries of the following tables (to within $\pm(6c)^{-1}T^{-\kappa-1}$ each):

$$\begin{aligned}&\left\{F_{v_{l,r},K_l}\left(\frac{n\pi}{\beta}\right) : l\in[0,m-1],\, r\in[0,M-1],\, \frac{n\pi}{\beta}\in[T-2c/\epsilon_1, T+T^\alpha+2c/\epsilon_1]\right\},\\ &\left\{F_{v_{m,r},K_m}\left(\frac{n\pi}{\beta}\right) : r\in[0,M'-1]\,,\ \frac{n\pi}{\beta}\in[T-2c/\epsilon_1\,,\, T+T^\alpha+2c/\epsilon_1]\right\},\\ &\left\{F_{v',K'}\left(\frac{n\pi}{\beta}\right) : \frac{n\pi}{\beta}\in[T-2c/\epsilon_1\,,\, T+T^\alpha+2c/\epsilon_1]\right\}.\end{aligned} \tag{2.19}$$

To compute the entries in the first table to within $\pm(6c)^{-1}T^{-\kappa-1}$ requires a number of operations of at most

$$\begin{aligned}&\sum_{l=0}^{m-1}\sum_{r=0}^{M-1} \lceil 10\beta(T^\alpha+4c/\epsilon_1)\rceil\, K_l \\ &\qquad \le \sum_{l=0}^{m-1}\sum_{r=0}^{M-1} 10(10+12c)2^l \le 150c\,2^m\,M \le 200c\,T^{1/2}\,,\end{aligned} \tag{2.20}$$

where the operations are carried out on numbers of $\lceil(10\kappa+10)\log T\rceil$ bits, say. Note the second inequality in (2.20) used the fact that $\beta/\epsilon_1 = 3$, which is true by construction. Similarly, the second and third formulas in (2.19) require at most $200c\,2^m\,M'$ and $200c\,2^m$ such operations, respectively. Since $M' \le M$, and $2^m M \le T^{1/2}$, then the total cost of precomputing the entries of all three formulas in (2.19) is less than

$$600c\,T^{1/2} \le 600(\kappa+2)\,T^{1/2}\log T \tag{2.21}$$

operations. Once the precomputation is done, the right side of (2.17) can be computed, with the aid of formula (2.16), to within $\pm T^{-\kappa-1}$ in less than

$$\begin{aligned}&20M + 20mM\lceil 2\beta c/\epsilon_1\rceil + 20M'\lceil 2\beta c/\epsilon_1\rceil + 20\lceil 2\beta c/\epsilon_1\rceil \\ &\qquad\qquad \le 1000(\kappa+2)\,T^\alpha(\log T)^2\end{aligned} \tag{2.22}$$

operations. Finally, the storage space requirement for precomputing the three formulas in (2.19) is at most

$$3mM\lceil 2\beta c/\epsilon_1\rceil\lceil(\kappa+1)\log T\rceil \le 1000(\kappa+2)^2\,T^{\alpha}(\log T)^3 \tag{2.23}$$

bits.

3. Comparison with the Riemann-Siegel formula

In this section, our algorithm is denoted by BLFI (band-limited function interpolation), and the Riemann-Siegel formula is denoted by RS.

The performance of the BLFI algorithm is compared with that of a relatively optimized version of the RS formula included in the *lcalc* library. The *lcalc* library is a C++ software developed by Michael Rubinstein to compute values of L-functions, including the zeta function. It can be downloaded at [R1].

The BLFI algorithm was also coded in C++, essentially as described in the previous sections[1]. We used double-precision arithmetic, which allows a maximum of 16 digits of accuracy[2].

We used the BLFI algorithm and the RS formula to evaluate $\zeta(1/2+it)$ on a grid of points of the form

$$[T\,, T+n\Delta]\,, \qquad n=1,\dots,N\,. \tag{3.1}$$

Here, $\Delta>0$ is the spacing (or density) of points[3], and N is the number of points in the grid. In case of the BLFI algorithm, an upper bound for the truncation error $\mathcal{E}$ was also chosen. The output of the precomputation needed by the BLFI algorithm is stored in dynamic memory, not saved in files.

The running times of the BLFI algorithm, presented in tables below, account for everything, including the precomputation. Also, when $T>10^{10}$, timings for the RS formula were obtained by evaluating zeta at a number of points $M<N$, then multiplying the running time by N/M.

Finally, our tests were carried out on a personal Macintosh dual 2.5 GHz PowerPC G5. The same compiler options were used for both programs.

Tables 2 and 3 list running times for BLFI and RS at various heights. The last column is the ratio of the time consumed by RS to the time consumed by the BLFI algorithm. At each height, we used a grid of 10^5 equidistant points. In Table 2, the spacing of the grid points is $\Delta=0.01$, and in Table 3 it is $\Delta=0.1$. The truncation error was chosen to satisfy $\mathcal{E}<10^{-8}$ in both tables.

Tables 2 and 3 indicate the running time of RS grows like $T^{1/2}$, as expected, while the running time of the BLFI algorithm grows roughly like $T^{1/4}$, also as expected. As T increases, the savings achieved by the BLFI algorithm are accentuated. For example, when $T=10^{16}$, and $\Delta=0.1$, the BLFI algorithm is more than 200 times faster than RS, but it is only 2 times faster when $T=10^8$.

---

[1]Our implementation of the BLFI algorithm differed from the description in the previous sections only in that we centered the band-limited functions $F_{v,K}$ so their spectrum is in $[-0.5\,\log(1+K/v), 0.5\,\log(1+K/v)]$ rather than $[0, \log(1+K/v)]$. This allows the recovery of values of the functions $F_{v,K}$ using less frequent sampling; see [O], pp. 90-91.

[2]Working with 30-digit arithmetic takes about 10 times longer for both RS and BLFI. But the slowdown can be made much less severe by following some of the tricks in [G] p. 13

[3]The grid of points where the zeta function is to be evaluated need not at all be uniform. What matters is the average spacing of the grid points.

TABLE 2. Running times of BLFI and RS with $N = 10^5$, $\Delta = 0.01$, and $\mathcal{E} < 10^{-8}$.

T	RS	BLFI	Ratio
10^8	0m 18s	0m 7s	2
10^{10}	2m 56s	0m 12s	14
10^{12}	29m 10s	0m 35s	50
10^{14}	348m 0s	2m 35s	134
10^{16}	3700m 0s	8m 28s	437

TABLE 3. Running times of BLFI and RS with $N = 10^5$, $\Delta = 0.1$, and $\mathcal{E} < 10^{-8}$.

T	RS	BLFI	Ratio
10^8	0m 18s	0m 9s	2
10^{10}	2m 56s	0m 27s	6
10^{12}	29m 10s	1m 34s	18
10^{14}	348m 0s	5m 50s	60
10^{16}	3700m 0s	18m 45s	205

We measure the sensitivity of the running time of the BLFI algorithm to perturbations in the values of its parameters and input. Table 4 indicates the running time of the BLFI algorithm grows roughly linearly with the number of grid points N, as soon as N is large enough, which is expected. Table 5 shows the running time is not radically affected by changes in the error allowance $\mathcal{E}$. This is not surprising either, because demanding higher precision increases the number of terms in the BLFI formula (2.14) only logarithmically.

TABLE 4. Running times of BLFI with $T = 10^{12}$, $\Delta = 0.1$, and $\mathcal{E} < 10^{-8}$.

N	BLFI
10^3	0m 2s
10^4	0m 10s
10^5	1m 34s
10^6	15m 27s

TABLE 5. Running times of BLFI with $T = 10^{12}$, $\Delta = 0.1$, and $N = 10^5$.

$\mathcal{E}$	BLFI
10^{-6}	1m 28s
10^{-8}	1m 34s
10^{-10}	1m 41s
10^{-12}	1m 48s
10^{-14}	1m 53s

Finally, Table 6 lists timings for the BLFI Algorithm for various values of Δ. As Δ increases, the grid expands, so more effort is exerted during the precomputation. This explains the slowdown in the method as Δ increases.

TABLE 6. Running times of BLFI with $T = 10^{12}$, $N = 10^5$, and $\mathcal{E} < 10^{-8}$.

Δ	BLFI
0.01	0m 35s
0.05	1m 10s
0.1	1m 34s
0.2	2m 8s
0.4	2m 54s

4. Appendix: band-limited interpolation

Much of the material in this section is contained in [O], pages 88-93. For the convenience of the reader, it is reproduced here with slight modifications. Let

$$G(t) = \int_{-\tau}^{\tau} g(x) e^{ixt} dx\,, \tag{4.1}$$

be a band-limited function, where $g(x)$ is a (finite) linear combination of delta functions supported on $(-\tau, \tau)$. It is well known that G can be recovered completely from its values at the grid points $\{n\pi/\tau \,|\, n \in \mathbf{Z}\}$. But this recovery process is not efficient, because it requires the values of G at many sample points $n\pi/\tau$.

The idea of a band-limited interpolation technique is to sample G on a denser grid, say $\{n\pi/\beta \,|\, n \in \mathbf{Z}\}$, where $\beta > \tau$, after which $G(t)$ can be recovered much more quickly, from its values at only a few grid points $n\pi/\beta$ that are close to t.

Specifically, choose $\beta > \tau$, and define $\lambda := (\beta+\tau)/2$, $\epsilon_1 := (\beta-\tau)/2$. Let I denote the characteristic function of the interval $[-\lambda, \lambda]$, and let H be any continuous function with total mass 1 supported on $[-\epsilon_1, \epsilon_1]$. Define $f := I * H$, the convolution of I and H, and let $\hat{f}$ and $\widehat{H}$ denote the Fourier transforms:

$$\hat{f}(t) := \int_{-\infty}^{\infty} f(x) e^{-itx}\, dx\,, \qquad \widehat{H}(t) := \int_{-\infty}^{\infty} H(x) e^{-itx}\, dx\,. \tag{4.2}$$

By construction, $f = I * H$ is supported on $[-\beta, \beta]$, and it is identically 1 on $[-\tau, \tau]$. And by hypothesis, $g(x)$ is supported on $[-\tau, \tau]$. Therefore, $f(x)g(x) \equiv g(x)$, the only difference is the left side involves smoothing in the "redundant" interval $[\tau, \beta] \cup [-\beta, -\tau]$. The latter observation is what gives the band-limited technique its edge. This is because some Fourier analysis yields,

$$\begin{aligned} G(t) &= \int_{-\infty}^{\infty} f(x) g(x) e^{ixt} = \frac{\lambda}{\beta} \sum_n G(n\pi/\beta)\, \hat{f}(t - n\pi/\beta) \\ &= \frac{\lambda}{\beta} \sum_{n=-\infty}^{\infty} G(n\pi/\beta) \frac{\sin \lambda(t - n\pi/\beta)}{\lambda(t - n\pi/\beta)} \widehat{H}(t - n\pi/\beta). \end{aligned} \tag{4.3}$$

In particular, we can try to choose the smoothing function $\widehat{H}$ so as to accelerate the convergence of the right side in (4.3). There are many choices for $\widehat{H}$ (e.g. a Gaussian). Following Odlyzko [O], we choose the following function, which still

closely resembles a Gaussian, but leads to smaller big-O constants (see [L] and [O], p. 92):

$$\widehat{H}(u) = \frac{c}{\sinh(c)} \frac{\sinh\sqrt{c^2 - \epsilon_1^2 u^2}}{\sqrt{c^2 - \epsilon_1^2 u^2}}\,, \tag{4.4}$$

where $c > 1$ is any fixed number. Finally, we only sum the terms with $|n\pi/\beta - t| < c/\epsilon_1$ in formula (4.3), which yields a truncation error $\mathcal{E}_1$ satisfying

$$|\mathcal{E}_1| < 2 \sup_{t\in\mathbb{R}} |G(t)| \int_{|u|>c/\epsilon_1} \left|\frac{\widehat{H}(u)}{u}\right| du. \tag{4.5}$$

This is bounded by (see [O], p. 92, and [L]):

$$|\mathcal{E}_1| < 2 \sup_{t\in\mathbb{R}} |G(t)| \log\frac{1+e^{-c}}{1-e^{-c}} \le 6 \sup_{t\in\mathbb{R}} |G(t)| e^{-c}\,. \tag{4.6}$$

Acknowledgements

The author would like to thank Michael Rubinstein for helpful discussions on the topic of this paper. The author would like to acknowledge the many helpful comments by the anonymous referee.

References

[Ga] W. Gabcke, *Neue Herleitung und explicite Restabschätzung der Riemann-Siegel-Formel.* Ph.D. Dissertation, Göttingen, 1979.

[G] X. Gourdon, *The 10^{13} first zeros of the Riemann zeta function and zero computation at very large heights.* Available at: http://numbers.compuation.free.fr.

[L] B.F. Logan, *Bounds for the tails of sharp-cutoff filter kernels.* SIAM J. Math. Anal. Volume 19, Issue 2, pp. 372-376 (March 1988). MR930033 (90m:94009)

[O] A.M. Odlyzko, *The 10^{20}-th zero of the Riemann zeta function and 175 million of its neighbors. www.dtc.umn.edu/∼odlyzko*

[OS] A.M. Odlyzko and A. Schönhage, *Fast algorithms for multiple evaluations of the Riemann zeta function.* Trans. Am. Math. Soc., Vol. 309, No. 2 (1988), 797-809. MR961614 (89j:11083)

[R] M.O. Rubinstein, *Computational methods and experiments in analytic number theory.* Recent Perspectives in Random Matrix Theory and Number Theory, London Mathematical Society, 2005, 425-506. MR2166470 (2006d:11153)

[R1] M.O. Rubinstein home-page, *www.math.uwaterloo.ca/∼mrubinst.*

Institute for advanced Study, 1 Einstein Drive, Princeton, New Jersey 08540

Current address: University of Waterloo, Department of Pure Mathematics, 200 University Avenue W, Waterloo, ON N2L 3G1, Canada

E-mail address: hiaryg@gmail.com