A nearly-optimal method to compute the truncated theta function, its derivatives, and integrals

By Ghaith Ayesh Hiary

Abstract

A poly-log time method to compute the truncated theta function, its derivatives, and integrals is presented. The method is elementary, rigorous, explicit, and suited for computer implementation. We repeatedly apply the Poisson summation formula to the truncated theta function while suitably normalizing the linear and quadratic arguments after each repetition. The method relies on the periodicity of the complex exponential, which enables the suitable normalization of the arguments, and on the self-similarity of the Gaussian, which ensures that we still obtain a truncated theta function after each application of the Poisson summation. In other words, our method relies on modular properties of the theta function. Applications to the numerical computation of the Riemann zeta function and to finding the number of solutions of Waring type Diophantine equations are discussed.

1. Introduction

Sums of the form

\[ \sum_{k=K_1}^{K_2} g(k) \exp(f(k)), \quad f(x) \in \mathbb{C}[x], \ g(x) \in \mathbb{C}[x] \]

arise in areas such as number theory, differential equations, lattice-point problems, optics, and mathematical physics, among others. For example, one encounters these sums in the context of Diophantine equations and fractional parts of polynomials ([Kor92]), solutions of heat and wave equations ([Mum83]), counting of integer points lying close to a curve ([Hux96]), numerical integration and quadrature formulas ([Kor92]), and motion of harmonic oscillators ([Kar04]). Due to the importance of such sums, there exists an abundance of methods to bound them. For instance, Vinogradov’s [Vin54] methods supply

---

Preparation of this material is partially supported by the National Science Foundation under agreements No. DMS-0757627 (FRG grant) and DMS-0635607 (while at the Institute for Advanced Study). This material is based on the author’s Ph.D. thesis.
such bounds, which along with some involved sieving techniques are used in attacking Goldbach-Waring type problems (see [LWY04] for example).

Despite the substantial interest in the sums (1.1), comparatively little is known about how to compute them for general values of their arguments. Yet in some settings, it is useful to be able to compute these sums efficiently and accurately. We soon describe two such settings, both of which originate in number theory.

The simplest examples of the exponential sums (1.1) occur when \( f(x) \) is of degree one, where we obtain the geometric series and its derivatives for which “closed-form” formulae are available. The first nontrivial example occurs when \( f(x) \) is a quadratic polynomial. In this case, the exponential sum (1.1) can be written as a linear combination of exponential sums of the form

\[
F(K,j; a,b) := \frac{1}{K^j} \sum_{k=0}^{K} k^j \exp(2\pi i ak + 2\pi i b k^2).
\]

Suppose the integer \( j \) is not too large. Then in this article, using ideas rooted in analysis, we prove the sum \( F(K,j; a,b) \) can be numerically computed to within \( \pm \varepsilon \), for any positive \( \varepsilon < e^{-1} \), in poly-log time in \( K/\varepsilon \). The linear and quadratic arguments \( a \) and \( b \) are any numbers in \([0, 1)\), and \( j \) is any integer that satisfies \( 0 \leq j \leq O((\log(K/\varepsilon))^{\kappa_0}) \), where \( \kappa_0 \) is any fixed constant.

More precisely, we obtain the following upper bound on the number of elementary arithmetic operations (additions, multiplications, evaluations of the logarithm of a positive number, or evaluations of the complex exponential) on numbers of \( O((j+1) \log(K/\varepsilon)) \) bits that our theta algorithm uses.

**Theorem 1.1.** There are absolute constants \( \kappa_1, \kappa_2, A_1, A_2, \) and \( A_3 \), such that for any positive \( \varepsilon < e^{-1} \), any integer \( K > 0 \), any integer \( j \geq 0 \), any \( a, b \in [0, 1) \), and with \( \nu := \nu(K, j, \varepsilon) = (j+1) \log(K/\varepsilon) \), the value of the function \( F(K,j; a,b) \) can be computed to within \( \pm A_1 \nu^{\kappa_1} \varepsilon \) using \( \leq A_2 \nu^{\kappa_2} \) arithmetic operations on numbers of \( \leq A_3 \nu^2 \) bits.

We remark that a bit complexity bound follows routinely from the arithmetic operations bound in Theorem 1.1. This is because all the numbers that occur in our algorithm have \( \leq A_3 \nu(K, j, \varepsilon)^2 \) bits. We do not try to obtain numerical values for the constants \( \kappa_1 \) and \( \kappa_2 \) in Theorem 1.1. With some optimization, they probably can be taken around 3. Also, in a practical version of the algorithm, the arithmetic can be performed using substantially fewer than \( A_3 \nu^2 \) bits, and we will likely be able to replace \( \nu(K, j, \varepsilon) \) with \( j + \log(K/\varepsilon) \). If we take \( \varepsilon = K^{-d} \) in the statement of the theorem, then \( \nu(K,j,\varepsilon) = (d+1)(j+1) \log K \). So the running time of the algorithm becomes \( \leq A_2 (d+1)^{\kappa_2} (j+1)^{\kappa_2} (\log K)^{\kappa_2} \) operations. For \( d \) and \( j \) bounded by any fixed power of \( \log K \), this running time is poly-log in \( K \).
We now discuss two applications of the algorithm of Theorem 1.1. For brevity, we will often refer to it as the “theta algorithm” because \( F(K, j; a, b) \) is directly related to the truncated theta function.

The values of \( \zeta(1/2 + it) \) on finite intervals are of great interest to number theorists. For example, the numerical verification of the Riemann hypothesis is clearly dependent on such data. There exist several methods to compute \( \zeta(1/2 + it) \), which means methods to obtain the numerical value of \( \zeta(1/2 + it) \) to within \( \pm t^{-\lambda} \), for any fixed \( \lambda > 0 \), and any \( t > 1 \). A well-known approach to computing \( \zeta(1/2 + it) \) relies on a straightforward application of the Riemann-Siegel formula. The Riemann-Siegel formula has a main sum of length \( \lfloor \sqrt{t/(2\pi)} \rfloor \) terms. A simplified version of that formula is

\[
\zeta(1/2 + it) = e^{-ib(t)} \Re \left( 2e^{-ib(t)} \sum_{n=1}^{n_1} n^{-1/2} \exp(it \log n) \right) + \Phi_\lambda(t) + O(t^{-\lambda}),
\]

where \( n_1 := \lfloor \sqrt{t/(2\pi)} \rfloor \), and \( \theta(t) \) and \( \Phi_\lambda(t) \) are certain well-understood functions that can be evaluated accurately in \( t^{\alpha(1)} \) operations on numbers of \( O_\lambda(\log t) \) bits; see [OS88]. (The notation \( O_\lambda(t) \) or \( t^{\alpha(1)} \) indicates asymptotic constants are taken as \( t \to \infty \), and they depend only on \( \lambda \), where we wish to compute \( \zeta(1/2 + it) \) to within \( \pm t^{-\lambda} \).

Our theta algorithm directly leads to a practical method to compute \( \zeta(1/2 + it) \) to within \( \pm t^{-\lambda} \) using \( t^{1/3+o(1)} \) operations on numbers of \( O_\lambda(\log t) \) bits and requiring \( O_\lambda(\log t) \) bits of storage. The derivation is explained in a general context in [Hia08] (similar manipulations can also be found in [Sch90] and in [Tit86, p. 99]). A preliminary step in the derivation is to apply appropriate subdivisions and Taylor expansions to the main sum in the Riemann-Siegel formula in order to reduce its computation to that of evaluating, to within \( \pm t^{-\lambda-1} \), a sum of about \( t^{1/3+o(1)} \) terms of the form \( F(K, j; a, b) \), where \( K = O(t^{1/6}) \), and \( 0 \leq j = O_\lambda(\log t) \). The power savings now follow because, using the theta algorithm, each of the functions \( F(K, j; a, b) \) can be evaluated to within \( \pm t^{-\lambda-2} \) in poly-log time in \( t \).

As another simple and direct application of the theta algorithm, we show how to find the number of solutions of a Waring type Diophantine equation. Suppose we wish to find the number of integer solutions to the system

\[
(1.4) \quad \sum_{r=1}^{s} (\alpha_r k_r + \beta_r k_r^2) - \sum_{r=s+1}^{s+t} (\alpha_r k_r + \beta_r k_r^2) \equiv 0 \pmod{M},
\]

where \( 0 \leq k_1, \ldots, k_{s+t} \leq K \), and \( \alpha_1, \beta_1, \ldots, \alpha_{s+t}, \beta_{s+t} \) are some fixed integers. A straightforward calculation reveals that the number of solutions is given by

\[
(1.5) \quad \frac{1}{M} \sum_{l=0}^{M-1} \left( \prod_{r=1}^{s} F(K, 0; \alpha_r l/M, \beta_r l/M) \right) \left( \prod_{r=s+1}^{s+t} F(K, 0; \alpha_r l/M, \beta_r l/M) \right).
\]
Using the theta algorithm, the expression (1.5) can be evaluated, to the nearest integer say, in $M^{1+o(1)}K^{o(s,t)}$ time. This is already significantly better than a brute-force search. One can also employ the fast Fourier transform to compute (1.5) with sufficient accuracy in $M^{1+o(1)}K^{o(s,t)} + K^{3+o(s,t)}$ time. But this is less efficient, and it requires temporarily storing large amounts of data. In the special case $M = K$, one can calculate (1.5) to the nearest integer in $M^{1+o(1)}K^{o(s,t)}$ time using well-known formulae for complete Gauss sums.

In searching for methods to compute $F(K, j; a, b)$, one should make use of the rich structure of the theta function. The theta function, together with variants, occurs frequently in number theory. It is directly related to the zeta function by a Mellin transform, and it has a functional equation as well as other modular properties. So one anticipates that a fast method to compute the truncated theta function will take advantage of this.

With this in mind, let us motivate the algorithm of Theorem 1.1 in the case $j = 0$. To this end, recall the following application of Poisson summation due to van der Corput (see [Tit86, p. 74], for a slightly different version). We refer to this application as the van der Corput iteration, although it is not conventionally labelled as such.

**Theorem 1.2** (van der Corput iteration). Let $f(x)$ be a real function with a continuous and strictly increasing derivative in $s \leq x \leq t$. Let $f'(s) = \alpha$ and $f'(t) = \beta$. Then

$$
(1.6) \quad \sum_{s \leq k \leq t} \exp(2\pi i f(k)) = \sum_{\alpha - \eta < v < \beta + \eta} \int_s^t \exp(2\pi i (f(x) - vx)) \, dx + R_{s,t,f},
$$

where $R_{s,t,f} = O(\log(2 + \beta - \alpha))$ for any positive constant $\eta$ less than 1.

The van der Corput iteration turns a sum of length $t - s$ terms into a sum of length about $\beta - \alpha = f'(t) - f'(s)$ terms, plus a remainder term $R_{s,t,f}$. In order for this transformation to be a potentially useful computational device, we need $\beta - \alpha \leq \tau(t - s)$ for some absolute constant $0 \leq \tau < 1$. This ensures that the new sum is shorter than the original sum. Moreover, we must be able to compute the remainder term $R_{s,t,f}$, and each of the integrals in the sum over $v$ in (1.6), using relatively few operations. For $\eta$ sufficiently small, the latter are precisely the integrals in the Poisson summation formula that contain a saddle point, where an integral is said to contain a saddle point if the function $\frac{d}{dv} (f(x) - vx) = f'(x) - v$ vanishes for some $x$ in the interval of integration $[s,t]$. So the integrals containing saddle points are determined by

$$
(1.7) \quad f'(x) = v, \quad \text{for some } x \in [s,t], \quad \iff \quad \alpha \leq v \leq \beta.
$$

Still, if we simply ensure $\beta - \alpha \leq \tau(t - s)$ for some fixed constant $0 \leq \tau < 1$, then the length of the sum over $v$ in (1.6) might be of the same order of
magnitude as the length of the original sum. For example, if \( \tau = 1/2 \), then we are only guaranteed a cut in the length by 1/2. So the complexity of the problem appears unchanged (in the sense of power-savings). But if we also require the function \( \exp(2\pi if(x)) \) to possess some favorable Fourier transform properties that allow us to turn the \( \nu \)-terms into ones suited for yet another application of the transformation (1.6), then under such hypotheses, one may hope repeated applications of the van der Corput iteration are possible. If they are, then one can compute the original sum over \( k \) using \( \leq \log_2 K \) applications of (1.6). (\( \log_2 x \) is the logarithm of \( x \) to base 2.)

These restrictions on \( f(x) \) and its Fourier transform are quite stringent. They severely limit the candidate functions for the proposed strategy. Fortunately, the choice \( f(x) = ax + bx^2 \), which occurs in \( F(K,j; a,b) \), is particularly amenable to repeated applications of the van der Corput iteration. Indeed, if we take \( s = 0 \) and \( t = K \) in relation (1.6), and assume \( [a] < [a+2bK] \) say, which is frequently the case, then with \( f(x) = ax + bx^2 \), and for \( \eta \) sufficiently small, the transformation (1.6) becomes

\[
\sum_{k=0}^{K} \exp(2\pi iak + 2\pi ibk^2) = \sum_{v=[a]}^{[a+2bK]} \int_0^K \exp(2\pi iax + 2\pi ibx^2 - 2\pi ivx) \, dx + R_1,
\]

where \( R_1 := R_1(a,b,K) \). We remark that if the condition \( [a] < [a+2bK] \) fails, so \( [a+2bK] \leq [a] \), then \( b < 1/K \). This means \( b \) will be relatively small. For such small \( b \), we will use the Euler-Maclaurin summation formula instead of the van der Corput iteration to calculate the sum on the left side in (1.8); see Section 3.2 for details. That aside, let us write the relation (1.8) as

\[
F(K; a, b) = \tilde{F}([a+2bK]; a, b) + R_1,
\]

where

\[
F(K; a, b) := \sum_{k=0}^{K} \exp(2\pi iak + 2\pi ibk^2),
\]

\[
\tilde{F}([a+2bK]; a, b) := \sum_{v=[a]}^{[a+2bK]} \int_0^K \exp(2\pi iax + 2\pi ibx^2 - 2\pi ivx) \, dx.
\]

We refer to sums of the form \( F(K; a, b) \) as quadratic sums. We recall the following “self-similarity” property of the Gaussian:

\[
\int_{-\infty}^{\infty} \exp(\eta t - t^2) \, dt = \sqrt{\pi} \exp(\eta^2/4), \quad \eta \in \mathbb{C}.
\]

With this setup, we describe the typical iteration of our algorithm. Using the identities in Lemma 4.1 in Section 4, as well as conjugation if necessary, it is easily shown the arguments \( a \) and \( b \) in (1.8) can always be normalized
so that \(a \in [0, 1)\) and \(b \in [0, 1/4]\). The normalization is important, otherwise successive applications of the Poisson summation (in the form of the van der Corput iteration) will essentially cancel each other. Since \(b \in [0, 1/4]\), the new sum \(\tilde{F}([a + 2bK]; a, b)\) has length \([a + 2bK] \leq K/2\), which is at most half the length of the original sum. We observe each term in \(\tilde{F}([a + 2bK]; a, b)\) is an integral of the form \(\int_0^K \exp(2\pi i a x + 2\pi i b x^2 - 2\pi i v x) \, dx\) for some \([a] \leq v \leq [a + 2bK]\). And by construction, each such integral contains a saddle-point. We extract the saddle point contribution from each of these integrals. To do so, we first shift the contour of integration to the stationary phase (at an angle of \(\pi/4\)). Then we complete the domain of integration on both sides to infinity. Last, we use the self-similarity of the Gaussian (1.11) to calculate the completed integral explicitly. This yields a new quadratic exponential sum \(\tilde{F}([a + 2bK]; a/2b, -1/4b)\). Slightly more explicitly, one obtains

\[
\tilde{F}([a + 2bK]; a, b) = e^{\pi i/4 - \pi a^2/(2b)} \sqrt{2b} \int_0^K \exp(2\pi i a x + 2\pi i b x^2 - 2\pi i v x) \, dx,
\]

where \(R_2 := R_2(a, b, K)\) is a remainder term. It is shown that the original remainder term \(R_1\) in (1.6), and the new remainder term \(R_2\) in (1.12), can both be computed to within \(\pm \varepsilon\) in poly-log time in \(K/\varepsilon\). Therefore, on repeating the typical iteration at most \(\log_2 K\) times, we arrive at a quadratic sum of a small enough length to be evaluated directly.

In the typical iteration, most of the effort is spent on computing the “error terms” \(R_1\) and \(R_2\). So in order for the overall algorithm to work, it is critical to prove that \(R_1\) and \(R_2\) can in fact be computed to within \(\pm \varepsilon\) in poly-log time in \(K/\varepsilon\). This is accomplished in detail in Sections 3 and 6. Briefly though, let us give a heuristic description of why that is.

The remainder terms \(R_1\) and \(R_2\) are implicitly defined by relations (1.8) and (1.12), respectively. It is not hard to show these definitions, together with the Poisson summation formula, and the self-similarity of the Gaussian, imply \(R_1\) and \(R_2\) must equal the following:

\[
R_1(a, b, K) = c_{a,b,K} + PV \sum_{v > [a + 2bK]} \int_0^K \exp(2\pi i a x + 2\pi i b x^2 - 2\pi i v x) \, dx,
\]

\[
R_2(a, b, K) = d_{a,b} + \sum_{v = [a]}^{[a + 2bK]} \int_{x < 0 \text{ or } x > K} \exp(2\pi i a x + 2\pi i b x^2 - 2\pi i v x) \, dx,
\]

where \(c_{a,b,K}\) and \(d_{a,b}\) are certain easily computable quantities, and PV in front of the sum in \(R_1\) means the terms of the infinite sum are taken in conjugate pairs. One observes none of the integrals in \(R_1\) and \(R_2\) contains a saddle point. Because, by construction, \(R_1\) consists of precisely the integrals in the Poisson summation formula with no saddle point, while \(R_2\) consists of “complements”
of such integrals, hence, by the monotonicity of \( \frac{d}{dx}(ax + bx^2 - vx) = a + 2bx - v \), they do not contain saddle points themselves.

The absence of saddle points from the geometric sums \( R_1 \) and \( R_2 \) is the reason they do not present any computational difficulty. This is because the absence of saddle points, when combined with suitable applications of Cauchy’s theorem, allows for their oscillations to be controlled easily and in an essentially uniform way. This means the same suitably chosen contour shift can be applied to a large subset of the integrals in \( R_1 \) (or \( R_2 \)) to ensure rapid exponential decay in the modulus of their integrands. The shifted integrals can thus be truncated quickly, and at a uniform point (after distance about \( \log(K/\varepsilon) \), where we wish to evaluated \( F(K; a, b) \) to within \( \pm \varepsilon \) say). Once truncated, the quadratic part of the integrand, which is \( \exp(2\pi ibx^2) \), can be expanded away as a polynomial in \( x \) of low degree (since \( 2\pi bx^2 \) no longer gets too large; see Section 3 and Lemmas 6.1 and 6.2 for the details). One then finds that in computing \( R_1 \) and \( R_2 \) the bulk of the computational effort is exerted on integrals of the form

\[
(1.13) \quad h(z, w) := \int_0^1 t^z \exp(wt) \, dt, \quad 0 \leq z, \ z \in \mathbb{Z}, \ \Re(w) \leq 0.
\]

The function \( h(z, w) \) is directly related to the incomplete gamma function. For purposes of our algorithm, the nonnegative integer \( z \) will be of size \( O(\log(K/\varepsilon)^\tilde{c}) \), where \( \tilde{c} \) is some absolute constant. In particular, the range of \( z \) is quite constrained, which enables a fast evaluation of the integrals (1.13) via relatively simple methods. But the literature is rich with methods to compute the incomplete gamma function, and consequently \( h(z, w) \), for general values of its arguments. These methods are surveyed in great detail by Rubinstein [Rub05], where they arise in the context of his derivation of a smoothed approximate functional equation for a general class of \( L \)-functions.

We further remark that the linear argument \( a \), and the quadratic argument \( b \), play different roles in the algorithm. Varying the linear argument \( a \) corresponds to sliding the sum over \( v \) in (1.8), whereas varying the quadratic argument \( b \) corresponds to compressing, or stretching, the sum. The latter feature greatly accounts for the utility of the van der Corput iteration in the context of the theta algorithm. Also, the role played by the self-similarity of the Gaussian is crucial, because it is the reason we still obtain a quadratic sum after each application of the van der Corput iteration, making its repetition natural to do.

At the beginning of each iteration, the algorithm normalizes the pair \((a, b)\) to be in \([0, 1) \times [0, 1/4] \). Afterwards, it computes the remainder terms \( R_1(a, b, K) \) and \( R_2(a, b, K) \) to within \( \pm \varepsilon \) in poly-log time in \( K/\varepsilon \). We comment that the remainder terms \( R_1 \) and \( R_2 \) can still be computed with the same accuracy and efficiency even if we only normalize \((a, b)\) to be in \([0, 1) \times [0, 1] \). However, the resulting quadratic sum, which is of length \( \approx 2bK \), could then be
longer the original sum, which is of length $K$. So, although normalizing $(a, b)$
to be in $[0, 1) \times [0, 1/4]$ is not important to computing the remainder terms $R_1$
and $R_2$ accurately and efficiently in a single iteration, it is important for the
recursive step in the algorithm.

Notice it is not enough to normalize the quadratic argument $b$ so it is
in $[0, 1/2)$ (this is straightforward to do using the periodicity of the complex
exponential and conjugation if necessary). Because if $b \in [0, 1/2)$, then $2bK$
could be very close to $K$. So the length of the new sum in the van der Corput
iteration, which is $\approx 2bK$, might be very close to the length of the original
sum, which is $K$. In particular, we will not have a sufficiently good upper
bound on the number of iterations required by theta algorithm. For example,
if $b$ starts close to $1/2$ mod 1, then its image under the map $b \mapsto -1/(4b)$,
which is the map that occurs in the algorithm, remains close to $1/2$ mod 1.
The extra ingredient needed to ensure that $b$ is bounded away from $1/2$
that in fact $b \in [0, 1/4]$, is the following (easily-provable) identity from Lemma 4.1:

\begin{equation}
(1.14) \quad F(K, j; a, b) = F(K, j; a \pm 1/2, b \pm 1/2) = F(K, j; a \mp 1/2, b \pm 1/2) .
\end{equation}

This concludes our sketch of the theta algorithm in the special case $j = 0$.
For a general $j \geq 0$, the theta algorithm consists of at most $\log_2 K$ iterations.
Each iteration acts on $F(K, j; a, b)$ in the following way:

\begin{equation}
(1.15) \quad F(K, j; a, b) = \sum_{l=0}^{j} w_{l, j, a, b, K} F\left(q_{a, b, K}, l; a^*_{a, b}, b^*_{a, b}\right) + R_{K, j, a, b},
\end{equation}

where $q_{a, b, K} := \lfloor a + 2bK \rfloor$, $a^*_{a, b} := a/(2b)$, $b^*_{a, b} := -1/(4b)$, and the coefficients $w_{l, j, a, b, K}$ are given by formula (6.14) in Lemma 6.3. The remainder term
$R_{K, j, a, b}$ is computed to within $\pm \varepsilon$ in poly-log time in $K/\varepsilon$, via the algorithm.

A key point is the tuple $(q_{a, b, K}, a^*_{a, b}, b^*_{a, b})$ does not depend on $j$. Therefore, the number of new sums $F(.)$ we need to compute in each iteration is always
$\leq j + 1$. And since the length of each new sum in (1.15) is $q_{a, b, K} \leq (K + 1)/2$,
the algorithm has to repeat at most $\log_2 K$ times.

More generally, our method acts on a sum of the form $\sum_{l=0}^{j} z_l F(K, l; a, b)$
in the following way:

\begin{equation}
(1.16) \quad \sum_{l=0}^{j} z_l F(K, l; a, b) = \sum_{l=0}^{j} \tilde{w}_{l, j, a, b, K} F\left(q_{a, b, K}, l; a^*_{a, b}, b^*_{a, b}\right) + \sum_{l=0}^{j} R_{K, l, j, a, b},
\end{equation}

where $q_{a, b, K}, a^*_{a, b},$ and $b^*_{a, b}$ are the same as in (1.15), and

\begin{equation}
(1.17) \quad \tilde{w}_{l, j, a, b, K} := \sum_{s=l}^{j} z_s w_{l, s, a, b, K} .
\end{equation}

In Section 3, we show that the coefficients $\tilde{w}_{l, j, a, b, K}$ do not grow too rapidly with
each iteration. Specifically, we show that the maximum modulus of $\tilde{w}_{l, j, a, b, K}$
over all iterations of the algorithm is $O(8^jK^2)$, provided the initial coefficients $z_l$ satisfy $\max_{0 \leq l \leq j} |z_l| = O(1)$ say, which is often the case. This bound is rather generous, but it is sharp enough for purposes of our error analysis, and for bounding the number of bits needed by the algorithm to perform its arithmetic operations.

The presentation is organized as follows. In Section 3, we describe the typical van der Corput iteration. In Section 4, we provide a pseudo-code for the algorithm. In Section 5, it is shown how to compute the related sums

\begin{equation}
G(K, j; a, b) := \sum_{k=1}^{K} \frac{1}{k^j} \exp(2\pi i ak + 2\pi ibk^2),
\end{equation}

with a similar complexity and accuracy to $F(K, j; a, b)$. This is done mainly for use in the separate paper [Hia08]. Finally, in Section 6, we give proofs of various lemmas employed in the previous sections. Section Section 6 includes Lemmas 6.6 and 6.7, which are also intended for use in the separate paper [Hia08]. These two lemmas give a complete account of how the theta algorithm behaves, in the case $j = 0$, under small perturbations in the linear argument $a$.

### 2. Notation

We let $[x]$ denote the largest integer less than or equal to $x$, $\lfloor x \rfloor$ denote smallest integer greater than or equal to $x$, $\{x\}$ denotes $x - \lfloor x \rfloor$, $\log x$ denote $\log_e x$, and $\exp(x)$ as well as $e^x$ stand for the exponential function (they are used interchangeably). We define $0^0 := 1$ whenever it occurs (e.g. in a binomial expansion). For easy reference, we list contours frequently used in later sections:

- $C_0 := \{t \mid 0 \leq t < K\}$, \hspace{1cm} $C_1 := \{K + it \mid 0 \leq t < K\}$,
- $C_2 := \{e^{\pi i/4} t \mid 0 \leq t < \sqrt{2}K\}$, \hspace{1cm} $C_3 := \{-it \mid 0 \leq t < \infty\}$,
- $C_4 := \{K - it \mid 0 \leq t < \infty\}$, \hspace{1cm} $C_5 := \{e^{\pi i/4} t \mid -\infty < t < 0\}$,
- $C_6 := \{e^{\pi i/4} t \mid \sqrt{2}K \leq t < \infty\}$, \hspace{1cm} $C_7 := \{e^{-\pi i/4} t \mid 0 \leq t < \sqrt{2}K\}$,
- $C_8 := C_2 \cup C_5 \cup C_6$, \hspace{1cm} $C_9 := \{t \mid 0 \leq t < \infty\}$.

Next, define the functions

\begin{equation}
I_C(K, j; a, b) := \frac{1}{K^j} \int_C t^j \exp(2\pi i at + 2\pi ibt^2) \, dt,
\end{equation}

\begin{equation}
J(K, j; M, a, b) := \frac{1}{K^j} \int_0^K t^j \exp \left( -2\pi at - 2\pi ibt^2 \right) \frac{1 - \exp(-2\pi Mt)}{\exp(2\pi t) - 1} \, dt.
\end{equation}

It is convenient to define $\tilde{I}_C(K, j; a, b) := I_C(K, j; ia, -b)$ because it will occur often. Notice $\tilde{I}_C(K, j; a, b) = e^{-\pi i/2}I_{e^{\pi i/2}C}(K, j; a, b)$, so it is essentially a
rotation by $\pi/2$. We also define

\[
\begin{align*}
p & := p(a) = \lceil a \rceil, & \omega_1 := \omega_1(a) = \lceil a \rceil - a, \\
q & := q(a, b, K) = \lceil a + 2bK \rceil, & \omega := \omega(a, b, K) = \{a + 2bK\}, \\
p_1 & := p_1(a, b, K) = q(a, b, K) - p(a), & \nu(K, l, \varepsilon) := (l + 1) \log(K/\varepsilon).
\end{align*}
\]

For any $j \geq 0$ and $\varepsilon \in (0, e^{-1})$, we say $K$ is large enough if it satisfies the lower bound $K > \Lambda(K, j, \varepsilon)$, where $\Lambda(K, j, \varepsilon) := 1000\nu(K, j, \varepsilon)^6$, and $\nu(K, j, \varepsilon) := (j + 1) \log(K/\varepsilon)$. For example, if $K$ is large enough, then among other consequences, $e^{-K} < (\varepsilon/K)^{1000(j+1)}$. Finally, in the remainder of the paper, any implicit asymptotic constants are absolute, and are applicable as soon as $\varepsilon < e^{-1}$, $0 \leq j$, and $\Lambda(K, j, \varepsilon) < K$, unless otherwise is indicated.

In Sections 3 and 4, we prove Theorem 1.1, which is our main theorem.

### 3. The basic iteration of the algorithm

Let $j$ be any nonnegative integer, $\varepsilon$ any number in $(0, e^{-1})$, $K$ any large enough integer, and $(a, b)$ any pair in $[0, 1) \times [0, 1)$ (the assumption $b \in [0, 1/4]$ is not needed in this section, but it is needed in §4). Then with $p := p(a) = \lceil a \rceil$, and $q := q(a, b, K) = \lceil a + 2bK \rceil$, either $p < q$ or $q \leq p$. The first possibility is the main case, and it is where the algorithm typically spends most of its time. The second possibility is a boundary point that will be handled separately using the Euler-Maclaurin summation.

#### 3.1. Main case: $p < q$

Let $p := p(K, a, b) = \lceil a \rceil - a$, and $q := q(K, a, b) = \lceil a + 2bK \rceil$. Assume $p < q$. By the Poisson summation formula

\[
F(K, j; a, b) = c_{bd} + \text{PV} \sum_{m=-\infty}^{\infty} IC_0(K, j; a - m, b),
\]

where $\delta_j$ is Kronecker’s delta, and $c_{bd} := c_{bd}(a, b, j, K) = \frac{1}{2} \delta_j + \frac{1}{2} e^{2\pi ibK + 2\pi j aK}$ is a boundary term. The notation PV in (3.1) stands for principal value, so terms are taken in conjugate pairs. Define

\[
\begin{align*}
S_1(K, j; a, b) & := \sum_{m=p}^{q} IC_0(K, j; a - m, b), \\
S_2(K, j; a, b) & := \text{PV} \sum_{m \notin [p, q]} IC_0(K, j; a - m, b).
\end{align*}
\]

Therefore,

\[
F(K, j; a, b) = c_{bd} + S_1(K, j; a, b) + S_2(K, j; a, b).
\]

Since the boundary term $c_{bd}$ in (3.3) can be computed in a constant number of operations on numbers of $O(\log K)$ bits, then it is enough to show how to deal with $S_1(K, j; a, b)$ and $S_2(K, j; a, b)$. We remark that the sum $S_1(K, j; a, b)$ corresponds to the terms in the Poisson summation formula that
contain a saddle point, and \( S_2(K, j; a, b) \) corresponds to the terms that do not contain a saddle point. The plan is to extract the saddle point contributions from \( S_1(K, j; a, b) \), which will yield a new (shorter) quadratic exponential sum, plus a remainder term (involving no saddle points) that we will show is computable to within \( \pm \varepsilon \) in poly-log time in \( K/\varepsilon \). As for \( S_2(K, j; a, b) \), whose terms do not contain saddle-points and hence will not contribute to the new quadratic sum, we will show it also can be computed in a similar amount of time and accuracy.

### 3.1.1. The sum \( S_1(K, j; a, b) \)

By definition,

\[
S_1(K, j; a, b) = \sum_{m=p}^{q} I_{C_0}(K, j; a - m, b),
\]

where \( p = \lceil a \rceil, q = \lfloor a + 2bK \rfloor \), \( C_0 := \{t \mid 0 \leq t < K\} \), and

\[
I_{C_0}(K, j; a - m, b) := \frac{1}{K^j} \int_{C_0} t^j \exp(2\pi i(a - m)t + 2\pi ibt^2) \, dt
\]

The integral \( I_{C_0}(K, j; a - m, b) \) contains a saddle-point when \( \frac{d}{dt}[(a - m)t - bt^2] \) vanishes for some \( 0 \leq t \leq K \), which is the interval of integration in (3.5). This occurs precisely when

\[
0 \leq (m - a)/(2b) \leq K \iff a \leq m \leq a + 2bK.
\]

Since (3.6) is exactly the range of summation in the definition of \( S_1(K, j; a, b) \), then each integral there contains a saddle-point. As stated earlier, we plan to extract the saddle-point contributions from these integrals, which will produce a new shorter quadratic exponential sum of length \( \leq q + 1 \) terms.

To this end, define the contours \( C_1 := \{K + it \mid 0 \leq t < K\} \), and \( C_2 := \{e^{\pi i/4}t \mid 0 \leq t < \sqrt{K}\} \). So \( C_1 \) and \( C_2 \) are the two other sides of a right-angle triangle with base \( C_0 \). By Cauchy’s theorem,

\[
S_1(K, j; a, b) = \sum_{m=p}^{q} I_{C_0}(K, j; a - m, b) - \sum_{m=p}^{q} I_{C_1}(K, j; a - m, b).
\]

We first consider the sum \( \sum_{m=p}^{q} I_{C_0}(K, j; a - m, b) \) in (3.7). Let us exclude the term corresponding to \( m = q \) in that sum for now as it will require a special treatment. We apply the change of variable \( t \leftarrow K + it \) to each integral \( I_{C_1}(K, j; a - m, b) \), followed by interchanging the order of summation, then
executing the resulting geometric sum, to obtain
\[
(3.8) \quad \sum_{m=p}^{q-1} I_{C_1}(K, j; a - m, b) = c_1 \sum_{l=0}^{j} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \frac{1}{K^l} \int_0^K t^l \exp \left( -2\pi \omega t - 2\pi ibt^2 \right) \times \frac{1 - \exp(-2\pi p_1 t)}{\exp(2\pi t) - 1} \, dt,
\]
where \( \omega = \{a + 2bK\} \), \( p_1 = q - p \), and \( c_1 := c_1(a, b, K) = ie^{2\pi i aK + 2\pi ibK^2} \). For any integer \( M \geq 0 \), define
\[
(3.9) \quad J(K, l; M, a, b) := \frac{1}{K^l} \int_0^K t^l \exp(-2\pi at - 2\pi ibt^2) \frac{1 - \exp(-2\pi Mt)}{\exp(2\pi t) - 1} \, dt.
\]
Then (3.8) can be expressed as
\[
(3.10) \quad \sum_{m=p}^{q-1} I_{C_1}(K, j; a - m, b) = c_1 \sum_{l=0}^{j} i^l \left( \begin{array}{c} j \\ l \end{array} \right) J(K, l; p_1, \omega, b).
\]

The integrand on the right side of (3.8) declines at least like \( e^{-2\pi t} \) with \( t \). The rapid decline permits the interval of integration to be truncated quickly, which enables an efficient evaluation of the (3.8), hence of \( J(K, l; p_1, \omega, b) \). (Notice if the term \( m = q \) were included in the sum (3.8), the integrand will decline only like \( e^{-2\pi \omega t} \), which might not be fast enough if \( \omega \) is very close to zero, and it is the reason that term was excluded earlier.)

Indeed, according to Lemma 6.1, which is proved via this approach, there exist absolute constants \( \kappa_3, \kappa_4, A_4, A_5, \) and \( A_6 \) such that \( J(K, l; p_1, \omega, b) \) can be evaluated (in terms of short exponential sums) to within
\[
\pm A_4 10^{\kappa_4} \nu(K, j, \varepsilon)^{\kappa_4} 4^{-j} \varepsilon
\]
using \( \leq A_5 10^{\kappa_4} \nu(K, j, \varepsilon)^{\kappa_4} \) operations on numbers of \( \leq A_6 \nu(K, j, \varepsilon)^2 \) bits. Notice the reason we built in the factor \( 4^{-j} \) in the accuracy is because each term in (3.10) is multiplied by \( \left( \begin{array}{c} j \\ l \end{array} \right) \leq 2^j \), and there are \( j + 1 \leq 2^j \) terms. But even if we require \( J(.) \) to be computed to within \( \pm 2^{-jd_1} K^{-d_2} \varepsilon \), say for any fixed \( d_1 \) and \( d_2 \), then the running time will still be polynomial in \( \nu(K, j, \varepsilon) \).

As for the term \( I_{C_1}(K, j; a - q, b) \), which we excluded earlier, it is treated as follows. Using the change of variable \( t \leftarrow K + it \), followed by a binomial expansion, we obtain
\[
(3.11) \quad I_{C_1}(K, j; a - q, b) = c_1 \sum_{l=0}^{j} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \frac{1}{K^l} \int_0^K t^l \exp(-2\pi \omega t - 2\pi ibt^2) \, dt
\]
\[
= c_1 \sum_{l=0}^{j} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \tilde{I}_{C_0}(K, l; \omega, b),
\]
where, as before, \( c_1 := c_1(a, b, K) = ie^{2\pi i aK + 2\pi ibK^2} \), and
\[
(3.12) \quad \tilde{I}_{C_0}(K, l; \omega, b) := \frac{1}{K^l} \int_0^K t^l \exp(-2\pi \omega t - 2\pi ibt^2) \, dt.
\]
The integrand in $\tilde{I}_{C_0}(K, l; \omega, b)$ might not experience rapid exponential decline, because $\omega$ could be very close to zero (recall $\omega := \{a + 2bK\}$, which could get arbitrarily close to zero). One overcomes this difficulty by using Cauchy’s theorem: let $C_7 := \{te^{-\pi i/4} | 0 \leq t < \sqrt{2K}\}$ and $C_1 := \{K - it | 0 \leq t < K\}$, so $C_7$ and $C_1$ are the two other sides of a right-angle triangle with base $C_0$; one finds

$$\sum_{l=0}^{\infty} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \tilde{I}_{C_0}(K, l; \omega, b) = \sum_{l=0}^{\infty} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \tilde{I}_{C_1}(K, l; \omega, b) - \sum_{l=0}^{\infty} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \tilde{I}_{C_1}(K, l; \omega, b).$$

The point now is that if $b$ is not too small, the functions $I_{C_7}(.)$ and $I_{C_1}(.)$ in (3.13) experience more rapid exponential decay, making them much easier to evaluate than $\tilde{I}_{C_0}(K, l; \omega, b)$. Specifically, by Lemma 6.1, each of the functions $\tilde{I}_{C_1}(K, l; \omega, b)$ and $\tilde{I}_{C_1}(K, l; \omega, b)$ can be evaluated to within

$$\pm A_7 \nu(K, j, \varepsilon) 8^{-j} \varepsilon$$

using $\leq A_8 \nu(K, j, \varepsilon) \nu$, provided $1 \leq 2bK$. Since in this subsection it is assumed $p = \lfloor a \rfloor < q = \lfloor a + 2bK\rfloor$, it follows $a + 1 \leq a + 2bK$, and so $1 \leq 2bK$. Put together, we have

$$\sum_{m=p}^{q} I_{C_1}(K, j; a - m, b) = c_1 \sum_{l=0}^{\infty} i^l \left( \begin{array}{c} j \\ l \end{array} \right) J(K, l; p_1, \omega, b)$$

$$+ c_1 \sum_{l=0}^{\infty} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \tilde{I}_{C_1}(K, l; \omega, b)$$

$$- c_1 \sum_{l=0}^{\infty} i^l \left( \begin{array}{c} j \\ l \end{array} \right) \tilde{I}_{C_1}(K, l; \omega, b),$$

where we have shown the right of (3.14) side can be computed accurately and efficiently enough for purposes of proving Theorem 1.1.

Having disposed of the sum of $\sum I_{C_1}(.)$ in (3.7), we now consider the sum $\sum I_{C_2}(.)$ there. Recall $C_2 := \{e^{\pi i/4} t | 0 \leq t < \sqrt{2K}\}$. We “complete” $C_2$ to span the full range $(-\infty, \infty)$. This yields $C_2 = C_8 - C_5 - C_6$, where $C_8 := \{e^{\pi i/4} t | -\infty < t < \infty\}$, $C_5 := \{e^{\pi i/4} t | -\infty < t < 0\}$, and $C_6 := \{e^{\pi i/4} t | \sqrt{2K} \leq t < \infty\}$. The advantage of rewriting $C_2$ in this way is the following. The integrand in

$$I_{C_2}(K, j; a - m, b) = \frac{e^{\pi i(j+1)/4}}{K^j} \int_{0}^{\sqrt{2K}} t^j \exp(2\pi i e^{\pi i/4}(a - m)t - 2\pi bt^2) dt,$$

experiences large oscillations that lead to a tremendous amount of cancellation. Consider, for instance, $|e^{2\pi i e^{\pi i/4}(a-m)t-2\pi bt^2}|$ reaches a maximum of
\[ e^{\pi(m-a)^2/(4b)} \] at the point \( 0 \leq t = (m-a)/(2\sqrt{2b}) \leq \sqrt{2K} \), while, in comparison, the actual value of the integral is typically much smaller in size. This makes \( I_{C_3}(\cdot) \) difficult to evaluate numerically for such \( m \). On the other hand, \( I_{C_8}(\cdot) \), which still involves a tremendous amount of cancellation, can be evaluated at once via formula (1.11), which is the self-similarity property of the Gaussian. Moreover, the extra integrals \( I_{C_5}(\cdot) \) and \( I_{C_6}(\cdot) \), which were needed to complete \( I_{C_2}(\cdot) \), can also be evaluated efficiently because when \( m \in \{p, \ldots, q\} \) and \( t \notin [0, \sqrt{2K}] \), the integrand \( e^{2\pi i e^{m/4}(a-m)t-2\pi b t^2} \) declines rapidly (a consequence of the lack a saddle-point there). Explicitly, since \( C_2 = C_8 - C_5 - C_6 \), we have

\[ I_{C_2}(K, j; a-m, b) = \sum_{m=p}^{q} I_{C_3}(K, j; a-m, b) - \sum_{m=p}^{q} I_{C_5}(K, j; a-m, b) - \sum_{m=p}^{q} I_{C_6}(K, j; a-m, b). \]

We consider \( \sum_{m=p}^{q} I_{C_5}(\cdot) \) first. Let us exclude the term corresponding to \( m = p \) from the sum for now because it will require a special treatment. By Cauchy’s theorem and a straightforward estimate, we obtain

\[ \sum_{m=p+1}^{q} I_{C_5}(K, j; a-m, b) = c_2 J(K, j; p_1, \omega_1, b) + O(e^{-K}), \]

where \( \omega_1 := [a] - a \), and \( c_2 := c_2(j) = (-1)^j e^{(j+1)\pi i/2} \). Like before, the integral \( J(\cdot) \) in (3.17) is handled by Lemma 6.1. To deal with the special term \( m = p \), we relate it to the integral \( \tilde{I}_{C_7}(\cdot) \):

\[ I_{C_5}(K, j; a-p, b) = c_2 \tilde{I}_{C_7}(K, j; \omega_1, b) + O(e^{-K}). \]

And we already know how to handle \( \tilde{I}_{C_7}(\cdot) \) via Lemma 6.2.

We now consider the sum \( \sum_{m=p}^{q} I_{C_6}(\cdot) \) in (3.16). It is not hard to see \( I_{C_6}(K, j; a-m, b) \) is bounded by \( O(e^{-K}/K) \) for each \( m = p, \ldots, q-1 \); hence, these terms are negligible due to our assumption \( K \) is large enough. And when \( m = q \), we have

\[ I_{C_6}(K, j; a-q, b) = c_3 2^{j+1} e^{-2\pi a K} \sum_{l=0}^{j} \binom{j}{l} \tilde{I}_{C_9}(K, l; \omega - i\omega + 2bK + i2bK, -2ib), \]

where \( c_3 := c_3(a, j, K) = e^{(j+1)\pi i/4 + 2\pi a K} \). The integral \( \tilde{I}_{C_9}(\cdot) \) above is also handled by Lemma 6.2. Finally, the sum \( \sum_{m=p}^{q} I_{C_8}(\cdot) \) in (3.16) produces the new quadratic exponential sums since by Lemma 6.3, we obtain

\[ \sum_{m=p}^{q} I_{C_8}(K, j; a-m, b) = \sum_{s=0}^{j} w_{s,j,a,b,K} F(q, s; a^*, b^*) - \delta_{1-p} w_{0,j,a,b,K}, \]
where $a^* \equiv a/(2b)(\text{mod } 1)$, $b^* \equiv -1/(4b)(\text{mod } 1)$, and as is apparent from formula (6.14) in Lemma 6.3, the coefficients $w_{s,j,a,b,K}$ can be computed to within $\pm 8^{-j}K^{-2}\varepsilon$ say for all $s = 0, 1, \ldots, j$ using $\leq A_{10} j^2$ operations on numbers of $\leq A_{11} \nu(K,j,\varepsilon)^2$ bits, where $A_{10}$ and $A_{11}$ are absolute constants.

More generally, if we wish to compute a linear combination of quadratic sums $\sum_{l=0}^j z_l F(K,l;a,b)$, rather than a single quadratic sum, then instead of (3.19), we obtain

\begin{equation}
\sum_{l=0}^j z_l \sum_{m=p}^q I_{C,s}(K,l;a-m,b) = \sum_{l=0}^j \tilde{w}_{l,j,a,b,K} F(q,l;a^*,b^*) - \delta_{1-p} \tilde{w}_{0,j,a,b,K},
\end{equation}

where

\begin{equation}
\tilde{w}_{l,j,a,b,K} := \sum_{s=l}^j z_s w_{l,s,a,b,K}.
\end{equation}

And we have the bound

\begin{equation}
|\tilde{w}_{l,j,a,b,K}| \leq \left( \max_{0 \leq l \leq j} |z_l| \right) \sum_{s=l}^j |w_{l,s,a,b,K}|.
\end{equation}

We consider the growth in the coefficients $\tilde{w}_{s,j,a,b,K}$ with each iteration. For our purposes, it suffices to bound the maximum modulus of $\tilde{w}_{s,j,a,b,K}$ over a full run of the algorithm. We examine two scenarios. In the first scenario, $2\nu(K,j,\varepsilon)^3 \leq 2bK$ say. Here, as a consequence of Lemma 6.4, we have (log$_2 K$ denotes the logarithm to the base 2):

\begin{equation}
\sum_{m=s}^j |w_{s,m,a,b,K}| \leq (2b)^{-1/2} e^{1+j/2\nu^3} (1 + j/\nu^3) \leq (2b)^{-1/2} e^{1+1/\log_2 K} (1 + 1/\log_2 K).
\end{equation}

In the second scenario, $2bK < 2\nu(K,j,\varepsilon)^3$. We observe this can happen only in the last iteration (because then $q = \lfloor a + 2bK \rfloor$ is not large enough, which is a boundary point of the algorithm). There are two possibilities: $q \leq p$ or $p < q < 2 \nu(K,j,\varepsilon)^3 + 1$. In the former case, the algorithm concludes via the Euler-Maclaurin summation method of Section 3.2, and not via the van der Corput iteration. In particular, if $q \leq p$, we do not reach the right side of (3.20) at all. In the latter case (the case $p < q < 2 \nu(K,j,\varepsilon)^3 + 1$), Lemma 6.4 supplies the bound

\begin{equation}
\sum_{m=s}^j |w_{s,m,a,b,K}| \leq (2b)^{-1/2} (j + 1)4^{j+2}.
\end{equation}

Therefore, by the bounds (3.22), (3.23), and (3.24), and taking into account the algorithm involves $\leq \log_2 K$ iterations and $1 \leq 2bK$, it follows that the
maximum modulus of the coefficients $\bar{w}_{s,j,a,b,K}$ that can occur over a full run of the algorithm
is
\begin{equation}
3.25 \quad \leq e^{\log_2 K (2 \log K)^2 (j+1)4^{j+2} \sqrt{K}} = O\left(8^j K^2\right).
\end{equation}

In Section 4, we use the bound (3.25) to determine by how much $\varepsilon$ needs
be adjusted over a full run of the algorithm so that the final output is still
accurate to within $\pm A_1 \nu^{\kappa_1} \varepsilon$, as claimed in Theorem 1.1.

3.1.2. The sum $S_2(K,j; a,b)$. By definition,
\begin{equation}
(3.26) \quad S_2(K,j; a,b) := \sum_{m=q+1}^{M} I_{C_0}(K,j; a-m,b) + \sum_{m=-M}^{p-1} I_{C_0}(K,j; a-m,b).
\end{equation}

Let us deal with the subsum $\sum_{m=q+1}^{M} I_{C_0}(K,j; a-m,b)$ first. If $m > q$, then
it holds
\begin{equation}
(3.27) \quad \left| I_{C_0-IT}(K,j; a-m,b) \right| \leq (2T)^j e^{-2\pi(1-\omega)T} \int_{0}^{K} e^{-4\pi b T (K-t)} dt \rightarrow_{T \rightarrow \infty} 0,
\end{equation}

where the fact $1 \leq 2bK$ was used to ensure $b$ is bounded from below. So by
Cauchy’s theorem we can replace $C_0$ with the contours $C_3 = \{ -it | 0 \leq t < \infty \}$
and $C_4 = \{ K - it | 0 \leq t < \infty \}$, which yields
\begin{equation}
(3.28) \quad \sum_{m=q+1}^{M} I_{C_0}(K,j; a-m,b) = \sum_{m=q+1}^{M} I_{C_3}(K,j; a-m,b) - \sum_{m=q+1}^{M} I_{C_4}(K,j; a-m,b).
\end{equation}

(We remark that if $j = 0$, then (3.27) holds uniformly in $a \in [0, 2]$ and integers
$m > q$. Therefore, (3.28) holds for all $a \in [0, 2]$ and integers $m > q = \lfloor a+2bK \rfloor$.
This observation is used in the proof of Lemma 6.6 later.) Now, by a routine
calculation
\begin{equation}
(3.29) \quad \sum_{m=q+1}^{M} I_{C_3}(K,j; a-m,b) = c_4 J(K,j; M-q, 2bK - \omega, b) + O(e^{-K}),
\end{equation}
where $c_4 := c_4(j) = e^{-(j+1)\pi i/2}$. A similar calculation gives
\begin{equation}
(3.30) \quad \sum_{m=q+2}^{M} I_{C_4}(K,j; a-m,b) = c_5 \sum_{l=0}^{j} (-i)^l \binom{j}{l} J(K,l; M-q-1, 1-\omega, b) + O(e^{-K}),
\end{equation}

where we isolated the term $I_{C_4}(K,j; a-q-1; b)$ since it will require a special
treatment, and where $c_5 := c_5(a,b,K) = -ie^{2\pi ia K + 2\pi ib K^2}$ (note $c_5 = -c_1$, where $c_1$ as in (3.8)). Last, in the case of $I_{C_4}(K,j; a-q-1; b)$, we have
\begin{equation}
(3.31) \quad I_{C_4}(K,j; a-q-1, b) = c_5 \sum_{l=0}^{j} (-i)^l \binom{j}{l} \tilde{I}_{C_0}(K,l; 1-\omega, b),
\end{equation}
A METHOD TO COMPUTE THE TRUNCATED THETA FUNCTION

where $C_9 := \{ t \mid 0 \leq t < \infty \}$. As before, the integrals $J(.)$ and $\tilde{I}_{C_9}(.)$, which occur in (3.29), (3.30) and (3.31), can by computed to within $\pm \varepsilon$ in polynomial time in $\nu(K, j, \varepsilon)$ by Lemmas 6.1 and 6.2.

As for the second subsum $\sum_{m=-M}^{p-1} I_{C_0}(K, j; a - m, b)$ in (3.26), the situation is analogous. We simply use the conjugates of the contours $C_3$ and $C_4$, then repeat the previous calculations with appropriate modifications, which results in the integrals

$$\sum_{m=-M}^{p-1} I_{C_3}(K, j; a - m, b) = c_7 \sum_{l=0}^{j} \frac{j!}{l!} J(K, l; M + p, 2bK - \omega_1, b) + O(e^{-K}),$$

and

$$\sum_{m=-M}^{p-1} I_{C_4}(K, j; a - m, b) = c_7 \sum_{l=0}^{j} \frac{j!}{l!} J(K, l; M + p, 2bK - \omega_1, b) + O(e^{-K}),$$

where $c_6 := c_6(j) = e^{(j+1)\pi i/2}$, and $c_7 := c_7(a, b, K) = ie^{2\pi iaK + 2\pi ibK^2}$ (note $c_7 = c_1$, where $c_1$ occurs in (3.8)). Once again, the functions on the right side in (3.32) and (3.33) can by computed to within $\pm \varepsilon$ in polynomial time in $\nu(K, j, \varepsilon)$ by Lemmas 6.1 and 6.2. Finally, the sum $\sum_{|m|>M} I_{C_0}(.)$ is bounded as follows:

$$\sum_{|m|>M} I_{C_0}(K, j; a - m, b) = \sum_{m>M} 2 \frac{K}{m} \int_{0}^{K} t^j \exp(2\pi iat + 2\pi ibt^2) \cos(2\pi mt) dt.$$  

Integrating by parts this is equal to

$$\sum_{m>M} \left( \frac{j}{\pi mK^j} \int_{0}^{K} (1 - \delta_j) t^{j-1} \exp(2\pi iat + 2\pi ibt^2) \sin(2\pi mt) dt + \frac{2i}{mK^j} \int_{0}^{K} t^{j} (a + 2bt) \exp(2\pi iat + 2\pi ibt^2) \sin(2\pi mt) dt \right).$$

By the second mean value theorem, we deduce for $M > 2K$ that

$$\sum_{|m|>M} I_{C_0}(K, j; a - m, b) = O \left( \sum_{m>M} \frac{K}{m(m-K)} \right) = O \left( \frac{K}{M} \right).$$

Finally, take $M = [8jK^3 e^{\nu(K, j, \varepsilon)}]$ to obtain

$$\sum_{|m|>M} I_{C_0}(K, j; a - m, b) = O(8^{-j} K^{-2} (\varepsilon/K)^{j+1}),$$

which suffices in light of our earlier bound (3.25) on the maximum modulus of the coefficients $\tilde{w}_{s,j,a,b,K}$ after a full run of the algorithm. We remark that one can let $M$ tend to $\infty$ unless $j = 0$, in which case, one can still let $M$ tend to
∞ provided the various $J(.)$ integrals are paired appropriately. Of course, this is what one should do in a practical implementation of the algorithm (we do not do this here to simplify the presentation).

In summary, we have shown the following: Let
\[ c_1 := i e^{2\pi i aK + 2\pi i bK^2}, \quad c_2 := (-1)^j e^{(j+1)\pi i/2}, \quad c_3 := e^{(j+1)\pi i/2 + 2\pi i aK}, \quad c_4 := e^{-(j+1)\pi i/2}, \quad c_5 := -i e^{2\pi i aK + 2\pi i bK^2}, \quad c_6 := e^{(j+1)\pi i/2}. \]

Let $w_{l,j} := w_{l,j,a,b,K}$ be defined as in (6.14), and let
\[ \tilde{c}_{bd} := \frac{1}{2} e^{2\pi i aK + 2\pi i bK^2} + \frac{1}{2} \delta_j - w_{0,j} \delta_{1-p}, \]
where $\delta_j$ is Kronecker’s delta. Also define
\[ a^* := a/(2b), \quad b^* := -1/(4b), \quad q := [a + 2bK], \quad \omega := \{a + 2bK\}, \quad \omega_1 := [a] - a, \quad p := [a], \quad p_1 := q - p. \]

Then, for $p < q$, $0 \leq j$, $\varepsilon \in (0, e^{-1})$, and $K$ large enough, it holds
\begin{equation}
(3.37) \quad F(K, j; a, b) = \sum_{l=0}^{j} w_{l,j} F(q, l; a^*, b^*) + \tilde{S}_1(K, j; a, b) + S_2(K, j; a, b) + \tilde{c}_{bd},
\end{equation}
where, for some absolute constant $\tilde{k}_1$, we have
\begin{equation}
(3.38) \quad \tilde{S}_1(K, j; a, b) = -c_1 \sum_{l=0}^{j} j! \frac{l^j}{l!} J(K, l; p_1, \omega, b) - c_2 J(K, j; p_1, \omega_1, b) - c_1 \sum_{l=0}^{j} j! \frac{l^j}{l!} \tilde{I}_{C_2}(K, l; \omega, b) + c_1 \sum_{l=0}^{j} j! \frac{l^j}{l!} \tilde{I}_{C_3}(K, l; \omega, b) - c_3 2^{j+1} e^{-2\pi \omega K} \sum_{l=0}^{j} \tilde{I}_{C_4}(K, l; \omega - i\omega + 2bK + i2bK, -2ib) - c_2 \tilde{I}_{C_2}(K, j; \omega_1, b) + O(\nu(K, j, \varepsilon)\tilde{k}_1 8^{-j} K^{-2\varepsilon}).
\end{equation}
\begin{equation}
(3.39) \quad S_2(K, j; a, b) = -c_5 \sum_{l=0}^{j} (-i)^l \frac{j!}{l!} J(K, l; M, 1 - \omega, b) + c_4 J(K, j; M, 2bK - \omega, b) + c_5 \sum_{l=0}^{j} j! \frac{l^j}{l!} J(K, l; M, 2bK - \omega_1, b) + c_6 J(K, j; M; 1 - \omega_1, b) - c_5 \sum_{l=0}^{j} (-i)^l \frac{j!}{l!} \tilde{I}_{C_5}(K, l; 1 - \omega, b) + c_6 \tilde{I}_{C_6}(K, j; 1 - \omega_1, b) + O(\nu(K, j, \varepsilon)\tilde{k}_1 8^{-j} K^{-2\varepsilon}).
\end{equation}
Furthermore, we have shown, with the aid of Lemmas 6.1 and 6.2, that each of the functions on the right side of (3.38) and (3.39) can be computed to within $O(\nu(K, j, \varepsilon)\tilde{\kappa}^2 8^{-j} K^{-2} \varepsilon)$ using $O(\nu(K, j, \varepsilon)\tilde{\kappa}^3)$ operations on numbers of $O(\nu(K, j, \varepsilon)^2)$ bits, where the constants $\tilde{\kappa}_2$ and $\tilde{\kappa}_3$ are absolute.

3.2. Boundary case: $q \leq p$. This occurs when $b$ is very small. We tackle it using the Euler-Maclaurin summation. Without loss of generality, one may assume $K$ is a multiple of 8. So we may write

$$F(K, j; a, b) = e^{2\pi i a K/2} + \frac{1}{K^j} \sum_{m=0}^{K-1} \sum_{k=mK/8}^{j} k^j \exp(2\pi iak + 2\pi ibk^2).$$

It suffices to deal with each inner sum in (3.40) since there are only eight of them. By a binomial expansion, we have

$$\frac{1}{K^j} \sum_{k=mK/8}^{j} k^j \exp(2\pi iak + 2\pi ibk^2) = c_{K, m} \sum_{l=0}^{\lfloor j/2 \rfloor} j^j m^j l ! \frac{1}{l !} F(K, l; a_{K, m}, b),$$

where $c_{K, m} = c_{K, m, a, b}$ is a quickly computable constant of modulus 1, $0 \leq m < 8$, $K_1 := K/8$, and $a_{K, m} := a_{K, m, a, b} = a + mbK/4$. Using the periodicity of the complex exponential, we can normalize $a_{K, m}$ so it satisfies $-1/2 \leq a_{K, m} \leq 1/2$. Since by assumption $q \leq p$, then $0 \leq a + 2bK < 2$. So $0 \leq 2bK < 2$, which implies $0 \leq a_{K, m} + 2bK < 1/4$. Therefore, $0 \leq |a_{K, m}| + 2bK < 3/4$. Put together, we may now assume our task is to compute a quadratic sum $F(K, j; a, b)$ with $|a| + |2bK| < 3/4$. To this end, define

$$f_{K, j, a, b}(t) := \frac{t^j}{K^j} \exp(2\pi i a t + 2\pi i b t^2).$$

By Lemma 6.5, we obtain

$$\max_{0 \leq t \leq K} |f_{K, j, a, b}^{(N)}(t)| \leq \left( \frac{j + N}{K} + 2\pi(|a| + |2bK|) \right)^N,$$

where $f_{K, j, a, b}^{(N)}(t)$ denotes the $N^{th}$ derivative with respect to $t$. Applying the Euler-Maclaurin summation formula to

$$(3.40) \quad F(K, j; a, b) = \frac{1}{K^j} \sum_{k=0}^{K} k^j \exp(2\pi iak + 2\pi ibk^2) =: \sum_{k=0}^{K} f_{K, j, a, b}(k),$$

yields

$$(3.41) \quad F(K, j; a, b) = \int_{0}^{K} f_{K, j, a, b}(t) \, dt + \sum_{n=0}^{N} \frac{(-1)^n B_n}{n!} (f_{K, j, a, b}^{(n-1)}(K) - f_{K, j, a, b}^{(n-1)}(0))$$

$$+ O\left( \frac{1}{N!} \int_{0}^{K} |B_N(\{t\})| f_{K, j, a, b}^{(N)}(t) \, dt \right),$$
where \{t\} denotes the fractional part of \(t\), \(B_n\) are the Bernoulli numbers, and \(B_n(t)\) are the Bernoulli polynomials; so \(B_0 = 1, B_1 = -1/2, B_2 = 1/6,\ldots\), and \(B_0(t) = 1, B_1(t) = t - 1/2, B_2(t) = t^2 - t + 1/6,\ldots\).

Taking \(N = [2 \log(8^3 K^3/\varepsilon)/\log(8/7) + 1]\) in (3.41), it follows from known asymptotics for \(B_n\) and \(B_n(t)\) (see [Rub05] for instance) that

\[
(3.42) \quad O \left( \frac{2}{(2\pi)^N} \int_0^K |f^{(N)}(t)| \, dt \right) = O \left( 2K(7/8)^{-N} \right) = O(8^{-j}K^{-2}\varepsilon). 
\]

Given our earlier bound (3.25) on the maximum modulus of the coefficients \(w_{s,t,a,b,K}\) after a full run of the algorithm, the bound (3.42) suffices for purposes of the algorithm.

Last, the correction terms in (3.41) can be computed quickly because there are only \(\leq N + 1 \leq 10\nu(K,j,\varepsilon)\) of them, and each can be computed to within \(\pm \varepsilon\) using \(O(\nu(K,j,\varepsilon^2))\) operations on numbers of \(O(\nu(K,j,\varepsilon)^2)\) bit via the recursion formula for \(f_{K,j,a,b}(t)\) provided in the proof of Lemma 6.5. It only remains to evaluate the integral \(\int_0^K f_{K,j,a,b}(t) \, dt\) in (3.41), which is the main term. But this is equal to \(I_{C_0}(K,j;a,b)\), which is handled by Lemma 6.2.

4. The Algorithm for \(F(K,j;a,b)\)

We call a real pair \((a, b)\) normalized if \((a, b) \in [0, 1) \times [0, 1/4]\). The normalization is important because sums are converted to integrals via Poisson summation. Therefore, different choices of \(a\) or \(b\) produce different integrals. We remark that it is mainly the normalization of quadratic argument \(b\) that truly matters. Normalizing \(a\) so that it is in the interval \([0, 1)\) is not critical to what follows. For example, it suffices to take \(a \in [-m, m]\) for a fixed integer \(m > 0\). To normalize the arguments \(a\) and \(b\) properly, we use the following lemma:

**Lemma 4.1.** For any integer \(K \geq 0\), any integer \(j \geq 0\), and any \(a, b \in \mathbb{C}\), the function \(F(K,j;a,b)\) satisfies the identities

\[
F(K,j;a,b) = F(K,j;a + 1,b) = F(K,j;a,b + 1)
\]

\[
= F(K,j;a \pm 1/2,b \pm 1/2) = F(K,j;a \mp 1/2,b \pm 1/2). 
\]

**Proof.** This follows from the fact \(\exp(2\pi i(z + 1)) = \exp(2\pi iz)\), and the fact \((k^2 \pm k)/2 \in \mathbb{Z}\) for any \(k \in \mathbb{Z}\). \(\square\)

As a direct application of the identities in Lemma 4.1, we obtain a simple procedure such that starting with any real pair \((a, b)\) it produces a normalized pair \((a_0, b_0) \in [0, 1) \times [0, 1/4]\) satisfying

\[
(4.2) \quad F(K,j;a,b) = F(K,j;a_0,b_0), \quad \text{or} \quad F(K,j;a,b) = \overline{F(K,j;a_0,b_0)}. 
\]
Notice that the pair \((a_0, b_0)\) is independent of \(K\) and \(j\). The normalization procedure is used in the pseudo-code below to compute \(\sum_{l=0}^{j} z_l F(K, l; a, b)\). As before, we let \(\nu(K, j, \varepsilon) := (j + 1) \log(K/\varepsilon)\), and \(\Lambda(K, j, \varepsilon) := 1000\nu(K, j, \varepsilon)^6\).

- **INPUT:** Numbers \(a, b \in [0, 1]\), an integer \(K > 0\), a positive number \(\varepsilon \in (0, e^{-1})\), an integer \(j \geq 0\), and an array of numbers \(z_l, l = 0, \ldots, j\), with \(|z_l| \leq 1\) say.

- **OUTPUT:** A complex number \(S\) that equals \(\sum_{l=0}^{j} z_l F(K, l; a, b)\) to within \(\pm A_1 \nu(K, j, \varepsilon)^{\kappa_1 \varepsilon}\), where \(A_1\) and \(\kappa_1\) are the absolute constants in Theorem 1.1.

- **INITIALIZE:** Set \(S = 0\), \(\text{flag} = 0\), and \(\text{counter} = 0\). It suffices to perform arithmetic using \(A_3 \nu(K, j, \varepsilon)^2\) bit numbers where \(A_3\) is the absolute constant in Theorem 1.1.

1. Normalize \((a, b) \leftarrow (a_0, b_0)\) using the identities in Lemma 4.1. This costs a constant number of operations on numbers of \(A_3 \nu(K, j, \varepsilon)^2\) bits. If conjugation is needed to normalize \((a, b)\), set flag \(\leftarrow 1\) and \(z_l \leftarrow z_l^*\).

2. Let \(p = \lfloor a_0 \rfloor,\) and \(q = \lfloor a_0 + 2b_0K \rfloor\). These numbers can be calculated using a constant number of operations on numbers of \(A_3 \nu(K, j, \varepsilon)^2\) bits.

3. If \(K < \Lambda(K, j, \varepsilon)\) (a boundary case), evaluate the sum \(\sum_{l=0}^{j} z_l F(K, l; a, b)\) directly. This can be done using \(\bar{A}_1 (j + 1) \nu(K, j, \varepsilon)\) operations on number of \(A_3 \nu(K, j, \varepsilon)^2\) bits, where \(\bar{A}_1\) is an absolute constant. Store the result in \(R_{\text{counter}}\). If flag = 1, set \(R_{\text{counter}} \leftarrow \overline{R_{\text{counter}}}\). Go to (9).

4. If \(q \leq p\) (a boundary case), apply the Euler-Maclaurin technique of Section 3.2 to evaluate the sum to within \(\tilde{\varepsilon}\) where \(\tilde{\varepsilon} := 8^{-J}K^{-2}\varepsilon\). This costs \(\bar{A}_2 \nu(K, j, \varepsilon)^{\tilde{k}_4}\) operations on numbers of \(A_3 \nu(K, j, \varepsilon)^2\) bits, where \(\bar{A}_2\) and \(\tilde{k}_4\) are absolute constants. (Notice \(\nu(K, j, \varepsilon) \leq 4(j + 1) \nu(K, j, \varepsilon)\), and so \(\bar{A}_3 \nu(K, j, \varepsilon)^{\tilde{k}_4} \leq 4^{\tilde{k}_4} \bar{A}_3 \nu(K, j, \varepsilon)^{2\tilde{k}_4}\).) Store the result in \(R_{\text{counter}}\). If flag = 1, set \(R_{\text{counter}} \leftarrow \overline{R_{\text{counter}}}\). Go to (9).

5. Apply the algorithm iteration for the case \(p < q\). This step requires the calculation of the quantities \(q := \lfloor a_0 + 2b_0K \rfloor, a^* := \frac{a_0}{\nu(K, j, \varepsilon)},\) and \(b^* := -\frac{1}{2\nu(K, j, \varepsilon)}\), all of which can be calculated using a constant number of operations. We obtain

\[
\sum_{l=0}^{j} z_l F(K, l; a, b) = \sum_{l=0}^{j} \tilde{w}_{l,j,a,b,K} F(q, l; a^*, b^*) + \sum_{l=0}^{j} R_{K,l,j,a,b},
\]

where \(\tilde{w}_{l,j,a,b,K} := \sum_{s=1}^{j} z_s w_{l,s,a,b,K}\). The remainder \(\sum_{l=0}^{j} R_{K,l,j,a,b,K}\) is computed by the algorithm to within \(\pm \bar{A}_4 \nu(K, j, \varepsilon)^{\kappa_5} \varepsilon\) using \(\leq A_5 \nu(K, j, \varepsilon)^{\kappa_6}\).
operations on numbers of $A_3 \nu(K, j, \varepsilon)^2$ bits, where $\tilde{A}_4$, $\tilde{A}_5$, $\tilde{\kappa}_5$, and $\tilde{\kappa}_6$, are absolute constants.

(6) Set $R[\text{counter}] = \sum_{l=0}^{j} R_l$, $z_l \leftarrow \sum_{s=l}^{j} z_{s,j} w_{l,s,a_0,b_0,K}$, $a \leftarrow a^*$, $b \leftarrow a^*$, $K \leftarrow q$, and $\text{counter} \leftarrow \text{counter} + 1$.

(7) If $\text{flag} = 1$, set $z_l \leftarrow z_l$, $R[\text{counter}] \leftarrow R[\text{counter}]$, $a \leftarrow -a$, $b \leftarrow -b$, and $\text{flag} \leftarrow 0$.

(8) Go to (1).

(9) Set $S = \sum_{l=0}^{\text{counter}} R[l]$. Return $S$.

5. The sums $G(K, j; a, b)$

We show how to evaluate the sums $G(K, j; a, b)$ defined in (1.18) to within $\pm \varepsilon$. Assume $K$ is large enough (i.e. $K > \Lambda(K, j, \varepsilon)$), otherwise we can evaluate the sum directly. Define

\[ \tilde{G}(N, j; a, b) := \sum_{k=N}^{2N-1} \frac{1}{k} \exp(2\pi i a k + 2\pi i b k^2). \]

It is not too hard to show that $G(K, j; a, b)$ can be written as the sum of $O(\log K)$ subsums of the form $\tilde{G}(N, j; a, b)$, with $N < K$, plus a remainder sum of length $O(\log K)$ terms. So it is enough to show how to compute $\tilde{G}(\cdot)$ to within $\pm \varepsilon$. Without loss of generality, we may assume $N$ is a multiple of 16, so we may write

\[ \tilde{G}(N, j; a, b) = \sum_{m=0}^{N_{m+1}-1} \sum_{k=N_m}^{k} \frac{1}{k} \exp(2\pi i a N_m k + 2\pi i b k^2), \]

where $N_m := N + mN/16$. The inner sum in the last expression is

\[ \frac{c_{N,m}}{N_m^j} \sum_{l=0}^{\infty} (-1)^l \binom{j + l - 1}{j - 1} \sum_{k=0}^{N_{m+1}-1} \frac{k^l}{N_m^l} \exp(2\pi i a_{N,m} k + 2\pi i b k^2), \]

where $c_{N,m} := c_{N,m,a,b}$ satisfies $|c_{N,m}| = 1$, and $a_{N,m} := a + 2bN_m$. Since $(j+l-1)k^l/N_m^{l+j} \leq 8^{-l}$, we can truncate the sum over $l$ in (5.3) after say $[10 \log(K/\varepsilon)]$ terms, which yields a truncation error of say $\pm \varepsilon/K$. Finally, by Theorem 1.1, each inner sum in (5.3) can be computed to within $\pm \varepsilon/K$, using $\leq 2^{\kappa_1} A_2 \nu(K, j, \varepsilon)^{\kappa_1}$ operations on numbers of $\leq A_3 \nu(K, j, \varepsilon)^2$ bits.

6. Auxiliary results

Lemma 6.1. There are absolute constants $\kappa_3$, $\kappa_4$, $A_4$, $A_5$, and $A_6$, such that for any positive $\varepsilon < e^{-1}$, any integer $0 \leq j$, any integer $10 \nu(K, j, \varepsilon)^2 < K$ say, any integer $0 < M < e^{10\nu(K, j, \varepsilon)^2}$ say, any $0 \leq w < K$ say, and any $0 \leq
b ≤ 1, the integral \( J(K, j; M, w, b) \) can be evaluated to within \( \pm A_4 \nu(K, j, \varepsilon)\kappa^3 \varepsilon \) using \( \leq A_5 \nu(K, j, \varepsilon)^{\kappa^4} \) operations on numbers of \( \leq A_6 \nu(K, j, \varepsilon)^2 \) bits.

Proof. The integrand in \( J(K, j; M, w, b) \) declines exponential fast, so the integral can be truncated quickly. Specifically, let \( L := L(K, j, \varepsilon) = \lceil \nu(K, j, \varepsilon) \rceil \); then

\[
J(K, j; M, w, b) = \frac{1}{K^j} \int_0^L t^j \exp\left(-2\pi w t - 2\pi i b t^2\right) \frac{1 - \exp(-2\pi M t)}{\exp(2\pi t) - 1} \, dt + O(\varepsilon) .
\]

Therefore, in order to evaluate \( J(K, j; M, w, b) \) in a time complexity as stated in the lemma, it suffices to deal with the integrals

\[
g(j, M, w, b, n) := \frac{1}{K^j} \int_{\nu}^{n+1} t^j \exp\left(-2\pi w t - 2\pi i b t^2\right) \frac{1 - \exp(-2\pi M t)}{\exp(2\pi t) - 1} \, dt ,
\]

where \( n \in \{0, \ldots, L - 1\} \). By the change of variable \( t \leftarrow t - n \), followed by Taylor expansions applied to the quadratic factor \( e^{-2\pi i b t^2} \), we obtain after some simple estimates that

\[
g(j, M, w, b, n) = \frac{\exp(-2\pi w n - 2\pi i b n^2)}{K^j} \sum_{s=0}^{j} \binom{j}{s} n^{j-s} \sum_{r=0}^{L} \frac{(-2\pi i b)^r}{r!} \times \int_0^1 t^{s+2r} \exp(-2\pi wt - 4\pi i b n t) \frac{1 - \exp(-2\pi M (t + n))}{\exp(2\pi (t + n)) - 1} \, dt + O(\varepsilon \log M) .
\]

Since the last expression is a linear combination of \( (L+1)(j+1) \leq 10 \nu(K, j, \varepsilon)^2 \) terms of the form

\[
(6.1) \quad \int_0^1 t^\alpha \exp(-2\pi wt - 4\pi i b n t) \frac{1 - \exp(-2\pi M (t + n))}{\exp(2\pi (t + n)) - 1} \, dt ,
\]

for integers \( 0 \leq \alpha \leq 2L + j \), then our task is reduced to dealing with the integral (6.1) over that range of \( \alpha \). To evaluate this integral, we first unfold the geometric series in the integrand; that is, we write (6.1) as

\[
(6.2) \quad \sum_{m=1}^{M} \exp(-2\pi mn) \int_0^1 t^{\alpha} \exp(-2\pi (m + w + 2ibn)t) \, dt .
\]

(Notice the integrals occurring in (6.2) are incomplete Gamma functions, which we alluded to earlier in formula (1.13). Although the methods given in this lemma to evaluate such integrals suffice for complexity bounds, there are other more practical, though more tedious to describe, methods.) Define \( m_{\alpha, n, w} := \max\{1, \lceil \alpha - w - 2bn \rceil \} \), in particular \( \alpha \leq m_{\alpha, n} + w + 2bn \). We split (6.2) into two subsums: \( \sum_{m_{\alpha, n} \leq m \leq M} \) and \( \sum_{1 \leq m < m_{\alpha, n}} \) (the splitting of the sum is because the general function \( h(z, w) := \int_0^1 t^z \exp(w t) \, dt \) behaves essentially differently according to whether \( |w| < |z| \) or \( |z| < |w| \)). Each term in the
subsum $\sum_{m, n}^{m, n} \leq m \leq M$ can be calculated explicitly as

$$\int_0^1 t^\alpha \exp(-2\pi(m + w + 2ibn)t) \, dt$$

$$= -\sum_{v=1}^{\alpha+1} \frac{\alpha!}{(\alpha + 1 - v)!} \frac{\exp(-2\pi m - 2\pi w - 4\pi ibn))}{(2\pi m + 2\pi w + 4\pi ibn)^v}$$

$$+ \frac{\alpha!}{(2\pi m + 2\pi w + 4\pi ibn)^{\alpha+1}}.$$  

(6.3)

So, on interchanging the order of summation, the subsum $\sum_{m, n}^{m, n} \leq m \leq M$ is equal to

$$-\sum_{v=1}^{\alpha+1} \frac{\alpha!}{(\alpha + 1 - v)!} \sum_{m=0}^{M} \frac{\exp(-2\pi mn) \exp(-2\pi m - 2\pi w - 4\pi ibn))}{(2\pi m + 2\pi w + 4\pi ibn)^v}$$

$$+ \alpha! \sum_{m=0}^{M} \frac{\exp(-2\pi mn)}{(2\pi m + 2\pi w + 4\pi ibn)^{\alpha+1}}.$$  

(6.4)

We claim expression (6.4) can be evaluated to within $\pm 100\nu(K, j, \varepsilon) \varepsilon$ using $\leq 1000\nu(K, j, \varepsilon)^2$ operations on numbers of $100\nu(K, j, \varepsilon)^2$ bits. To see why, notice if $n \neq 0$, the series over $m$ can be truncated after $L := L(K, j, \varepsilon)$ terms, with a truncation error $\leq 10(\alpha + 1) \exp(-2\pi n(\alpha + L)) \leq 10 \varepsilon$, where we used the facts $\alpha^v \leq (m_{\alpha,n} + w + 2bn)^v$, which holds by construction, and $\alpha!/(\alpha + 1 - v)! \leq \alpha^v$. Once truncated, the series (6.4) can be evaluated directly in $\leq 100L(K, j, \varepsilon)$ operations. If $n = 0$, the series (6.4) is equal to

$$-\sum_{v=1}^{\alpha+1} \frac{\alpha!}{(\alpha + 1 - v)!} \sum_{m=0}^{M} \frac{\exp(-2\pi mn) \exp(-2\pi m - 2\pi w)}{(2\pi m + 2\pi w)^v}$$

$$+ \alpha! \sum_{m=0}^{M} \frac{1}{(2\pi m + 2\pi w)^{\alpha+1}}.$$  

(6.5)

Since the terms in the first series over $m$ in (6.5) decline exponentially fast with $m$ (due the the decay provided by the term $e^{-2\pi m}$), it can be truncated early, after $L := L(K, j, \varepsilon)$ terms, with truncation error $\leq 10 \varepsilon$. The truncated series can then be evaluated directly. As for the second series in (6.5), it can be calculated efficiently using the Euler-Maclaurin summation formula; specifically, the initial sum $\sum_{m,n}^{m, n} \leq m < 10(m_{\alpha,n} + L)$, which consists of $\leq 10(m_{\alpha,n} + L) \leq 100 \nu(K, j, \varepsilon)$ terms, is evaluated directly, while the tail sum $\sum_{10(m_{\alpha,n} + L) \leq m \leq M}^{10} \nu(K, j, \varepsilon)$ terms is evaluated to within $\pm 10 \nu(K, j, \varepsilon)$ using an Euler-Maclaurin formula like (3.41) at a cost of $\leq 100 \nu(K, j, \varepsilon)^2$ operations on numbers of $\leq 100 \nu(K, j, \varepsilon)^2$ bits say.

It remains to deal with the subsum $\sum_{1 \leq m < m_{\alpha,n}}$ from (6.2). Since this subsum consists of $< m_{\alpha,n} = 2L + j \leq 10 \nu(K, j, \varepsilon)$ terms, it suffices to show how to deal with a single term there, which is essentially an integral of the
say that for any positive 0 \leq \alpha \leq L of degree 2

\begin{equation}
(6.6) \quad \int_0^1 t^\alpha \exp \left(-2\pi(m + w + 2ibn)t\right) dt, \quad 1 \leq m < m_{\alpha,n}.
\end{equation}

To do so, we apply the change of variable \( t \leftarrow [m + w + 2bn]t \) to (6.6) to reduce it to a sum of the \([m + w + 2bn] \leq 10 \nu(K, j, \varepsilon)\) integrals

\begin{equation}
(6.7) \quad \frac{1}{[m + w + 2bn]^\alpha+1} \int_l^{l+1} t^\alpha \exp \left(-2\pi(m + w + 2ibn)t/[m + w + 2bn]\right) dt,
\end{equation}

where 0 \leq l \leq [m + w + 2bn] - 1 is an integer. The integrals (6.7) are straightforward to evaluate: one makes the change of variable \( t \leftarrow t - l \), then uses Taylor expansions to break down the integrand into a polynomial in \( t \) of degree \( 2L + \alpha \) say, plus an error of size \( O(\varepsilon) \), and finally one integrates each term explicitly (note each term is just a monomial \( z^d \) for some integer \( 0 \leq d \leq 2L + \alpha \), and some quickly computable coefficient \( z_d \)).

**Lemma 6.2.** There are absolute constants \( \kappa_5, \kappa_6, A_7, A_8, \) and \( A_9 \), such that for any positive \( \varepsilon < e^{-1} \), any integer \( 0 \leq j \), any integer \( 10 \nu(K, j, \varepsilon)^2 < K \) say, any \( 0 \leq b \leq 1 \) satisfying \( 1 \leq 2bK \) say, and any \( 0 \leq w \leq 1 \) say, each of the integrals

\begin{align*}
\bar{I}_{C_1}(K, j; w, b), & \quad \bar{I}_{C_7}(K, j; w, b), \\
\bar{I}_{C_9}(K, j; w, b), & \quad \bar{I}_{C_9}(K, j; w - iw + 2bK + i2bK, -2ib)
\end{align*}

can be evaluated to within \( \pm A_7 \nu(K, j, \varepsilon)^{\kappa_5} \varepsilon \) using \( A_8 \nu(K, j, \varepsilon)^{\kappa_6} \) operations on numbers of \( \leq A_9 \nu(K, l, \varepsilon)^2 \) bits. Moreover, under the same assumptions on \( K, j, \) and \( b \), as above, except \( b \) need not satisfy the condition \( 1 \leq 2bK \), and for any \(-1 \leq a \leq 1 \) say, the integral \( I_{C_9}(K, j; a, b) \) can be evaluated with the same accuracy and efficiency as the above four integrals.

**Proof.** We show how to compute \( \bar{I}_{C_1}(K, j; w, b) \) first. We have

\begin{equation}
\bar{I}_{C_1}(K, j; w, b) = c_8 e^{-2\pi w K} \sum_{l=0}^{j} \left( \frac{j}{l} \right) \frac{(-i)^l}{K^l} \int_0^K t^l \exp \left(2\pi iwt - 4\pi bKt + 2\pi ib^2t^2\right) dt,
\end{equation}

where \( c_8 := c_8(b, K) = -ie^{-2\pi ibK^2} \). Since \( 2bK \geq 1 \) by hypothesis, we can truncate the interval of integration above at \( L := L(K, j, \varepsilon) = [\nu(K, j, \varepsilon)] \), which reduces our task to evaluating \( (j + 1)L \) integrals of the form

\begin{equation}
(6.8) \quad \frac{1}{L^j} \int_n^{n+1} t^l \exp \left(2\pi iwt - 4\pi bKt + 2\pi ib^2t^2\right) dt,
\end{equation}

for integers \( 0 \leq l \leq j \) and \( 0 \leq n \leq L - 1 \). To evaluate (6.8), substitute \( t \leftarrow t - n \), then eliminate the quadratic term \( \exp(2\pi ib^2t^2) \) using Taylor expansion. This
results in a linear combination, with quickly computable coefficients each of size \(O(1)\), of, say, 3L integrals of the form

\[
\int_0^1 t^\alpha \exp(2\pi i nt) \, dt,
\]

where \(\eta := \eta_{n,w,b,K} = w + 2bn + 2ibK\) and \(0 \leq \alpha < 3L\) an integer. The integrals (6.9) are easily-calculable: if \(\alpha < |w + 2bn + 2ibK|\), we evaluate (6.9) explicitly as was done in (6.3), and if \(|w + 2bn + 2ibK| \leq \alpha\), we follow similar techniques to those used to arrive at expression (6.7) earlier. The evaluation of \(\tilde{I}_{C_{n}}(K, l; w - iw + 2bK + i2bK, -2ib)\) is completely similar to \(\tilde{I}_{C_{n}}(K, j; w, b)\), already considered.

We move on to \(\tilde{I}_{C_{j}}(K, j; w, b)\). We have by definition

\[
\tilde{I}_{C_{j}}(K, j; w, b) = \frac{c_0}{K^j} \int_0^{\sqrt{2}K} t^j \exp \left(-\sqrt{2}\pi wt + \sqrt{2}\pi iwt - 2\pi bt^2 \right) \, dt,
\]

where \(c_0 := c_0(j) = \exp(- (j + 1)\pi i/4)\). The change of variable \(t \leftarrow \sqrt{b} t\) yields

\[
\tilde{I}_{C_{j}}(K, j; w, b) = \frac{c_0}{b^{(j+1)/2}K^j} \int_0^{\sqrt{2}K} t^j \exp \left(-2\pi \frac{w}{\sqrt{2}b} t + 2\pi i \frac{w}{\sqrt{2}b} t - 2\pi t^2 \right) \, dt.
\]

So, truncating the interval of integration at \(\lceil \sqrt{L} \rceil\) reduces the problem to evaluating

\[
\frac{c_0}{b^{(j+1)/2}K^j} \int_n^{n+1} t^j \exp \left(-2\pi \frac{w}{\sqrt{2}b} t + 2\pi i \frac{w}{\sqrt{2}b} t - 2\pi t^2 \right) \, dt,
\]

for integers \(0 \leq n < \lceil \sqrt{L} \rceil\). The integrals are handled as follows: substitute \(t \leftarrow t - n\), then eliminate the quadratic term using Taylor expansions, this results in integrals similar to (6.9), which we already know how to handle.

Next, we consider \(\tilde{I}_{C_{n}}(K, j; w, b)\). If \(w = 0\), this integral is quickly calculable via the self-similarity formula (1.11), or some variation of it. So we may assume \(w > 0\). Since

\[
\left| \frac{1}{K^j} \int_0^T (T - it)^j \exp \left(-2\pi w(T - it) - 2\pi ib(T - it)^2 \right) \, dt \right| \to_{T \to \infty} 0,
\]

then by Cauchy’s theorem, we may replace \(C_0\) by \(e^{-\pi i/4}C_0\) in \(\tilde{I}_{C_{n}}(K, j; w, b)\). (We remark that if \(j = 0\), then (6.11) holds uniformly in \(0 \leq \omega \leq 1\). This observation is used in the proof of Lemma 6.6 later.) Combined with a straightforward estimate, this yields

\[
\tilde{I}_{e^{-\pi i/4}C_0}(K, j; w, b) = \tilde{I}_{C_{j}}(K, j; w, b) + O(e^{-K}),
\]

which we have already shown how to compute.

Last, we consider the integral \(I_{C_{n}}(K, j; a, b)\). This may contain a critical point or it may not according to whether \(-a/(2b) \in [0, K]\) or not. We supplied methods to deal with these possibilities in Sections 3.1.1 and 3.1.2 respectively, provided \(1 \leq 2bK\). But the same methods still apply as long as \(b\) is not too small, say \(1 < bK^2\). If not, say \(b < 1/\sqrt{K}\), then computing \(I_{C_{n}}(K, j; a, b)\)
is straightforward anyway because one can apply Taylor expansions to the quadratic factor \( \exp(2\pi ib^2) \) in \( I_{C_a}(K, j; a, b) \) to reduce it to a polynomial in \( t \) of degree \( 2L \) say, plus an error of size \( O(\varepsilon) \), which, on applying the change of variable \( t \leftarrow t/K \), yields an integral similar to (6.9), which we have already shown how to handle.

\[ I_{C_a}(K, j; a - m, b) = \exp \left( \frac{2\pi i a}{2b} - \frac{2\pi i}{4b} m \right) \sum_{s=0}^{j} \frac{w_{s,j,a,b,K} m^s}{q^s} , \tag{6.13} \]

\[ w_{s,j,a,b,K} = q^s j! \sqrt{2\pi e^{\pi i/4} e^{(j-s)3\pi i/4} e^{-i\pi a^2/(2b)}} \left( \frac{2\pi}{b} \right)^{j-s} \left( \frac{ae^{3\pi i/4}}{\sqrt{2\pi / b}} \right)^l. \tag{6.14} \]

**Lemma 6.3.** For any integer \( K > 0 \), any integer \( j \geq 0 \), any integer \( m \), any \( a \in \mathbb{R} \), and any \( b > 0 \) such that \( q := [a + 2bK] \) is not zero, we have

\[ \sum_{m=s}^{0} |w_{s,m,a,b,K}| \leq e \sqrt{2b} \left( 1 + \frac{1}{2bK} \right)^j \sum_{g=0}^{j} \left( \frac{j}{2bK} \right)^g. \tag{6.15} \]

If, in addition, \( 2bK \leq 4 \nu(K, j, \varepsilon)^3 \) say, then

\[ \sum_{m=s}^{j} |w_{s,m,a,b,K}| \leq (2b)^{-1/2}(j + 1)4^{j+2}. \]

**Proof.** This follows from well-known properties of the Hermite polynomials; see [Ism05]. \( \square \)

**Lemma 6.4.** For any \( \varepsilon \in (0, e^{-1}) \), any \( a \in [0, 1] \), any \( b \in [0, 1] \), any integer \( j \geq 0 \), any positive integer \( K > \Lambda(K, j, \varepsilon) \), any integer \( 0 \leq s \leq j \), let \( w_{s,m,a,b,K} \) be defined as in (6.14), then assuming \([a] < [a + 2bK]\), we have

\[ \sum_{m=s}^{j} |w_{s,m,a,b,K}| \leq \left( 1 + \frac{a}{2bK} \right)^j \frac{1}{\sqrt{2b}} \left[ \sum_{g=0}^{j} \left( \frac{j}{2bK} \right)^g \right] e^a. \]
The bound (6.15) now follows because \( a \in [0, 1] \) by hypothesis. To prove the last part of the lemma, notice if \( 2bK \leq 4v(K, j, \varepsilon)^3 \), then since \( \Lambda(K, j, \varepsilon) < K \), it follows \( b < 1/(2j + 2)^2 \). Also, the assumption \( |a| < |a + 2bK| \) implies \( 1/(2K) \leq b \). Therefore, by the definition (6.14), and a direct calculation,

\[
(6.16) \quad \frac{\sum_{m=0}^{j} w_{s,m,a,b,K}}{\sqrt{2b}} \leq \frac{2q^s}{(2bK)^s} \sum_{m=0}^{j-s} \frac{(m+s)!}{s! m! (2bK)^m} \leq \frac{(j+1)^{4j+2}}{\sqrt{2b}}. \]

□

**Lemma 6.5.** For any integer \( j \geq 0 \), any integer \( m \geq 0 \), any integer \( K > 0 \), and any real numbers \( a \) and \( b \), the function

\[
f_{K,j,a,b}(x) := \frac{x^j}{K^j} \exp(2\pi i a x + 2\pi i b x^2)
\]

satisfies

\[
(6.17) \quad \max_{0 \leq x \leq K} |f_{K,j,a,b}(x)| \leq (2\pi(|a| + |2bK|) + (m + j)/K)^m.
\]

Proof. \( f_{K,j,a,b}(x) = P_{m,K,j,a,b}(x) \) \exp(2\pi i a x + 2\pi i b x^2), where \( P_{m,K,j,a,b}(x) \) is a polynomial in \( x \) of degree \( m + j \). So \( P_{m,K,j,a,b}(x) := \sum_{l=0}^{m+j} d_{l,m,K,j,a,b} x^l \) for some coefficients \( d_{l,m,K,j,a,b} \) defined by the recursion

\[
(6.18) \quad P_{m+1,K,j,a,b}(x) = 2\pi i (a + 2bx)P_{m,K,j,a,b}(x) + P'_{m,K,j,a,b}(x),
\]

where \( P_{0,K,j,a,b}(x) := x^j/K^j \) and \( P'_{m,K,j,a,b}(x) \) is the derivative of \( P_{m,K,j,a,b}(x) \) with respect to \( x \). Notice \( |f_{K,j,a,b}(x)| = |P_{m,K,j,a,b}(x)| \). Define

\[
|P_{m,K,j,a,b}(x)|_1 := \sum_{l=0}^{m+j} |d_{l,m,K,j,a,b} x^l|
\]

and notice \( |P(x)| \leq |P(x)|_1 \). By induction on \( m \), suppose

\[
(6.19) \quad \max_{0 \leq x \leq K} |P_{m,K,j,a,b}(x)|_1 \leq (2\pi(|a| + |2bK|) + (m + j)/K)^m.
\]

Clearly, (6.19) holds when \( m = 0 \), and it is straightforward to verify

\[
(6.20) \quad \max_{0 \leq x \leq K} |P'_{m,K,j,a,b}(x)|_1 \leq \frac{m + j}{K} \max_{0 \leq x \leq K} |P_{m,K,j,a,b}(x)|_1.
\]

On combining relations (6.18) and (6.20), we obtain

\[
(6.21) \quad \max_{0 \leq x \leq K} |P_{m+1,K,j,a,b}(x)|_1 \leq \max_{0 \leq x \leq K} |2\pi i (a + 2bx)P_{m,K,j,a,b}(x)|_1
\]

\[
+ \max_{0 \leq x \leq K} |P'_{m,K,j,a,b}(x)|_1 \leq (2\pi(|a| + |2bK|) + (m + 1 + j)/K)^{m+1},
\]

as required. Notice the inductive proof naturally gives a method to compute the polynomials \( P_{m,K,j,a,b}(x) \).

□

**Lemma 6.6.** Let \( \varepsilon \in (0, e^{-1}) \), \( a \in [0, 2] \), \( b \in [0, 1/4] \), and \( K > 0 \) an integer. Define \( \nu(K, \varepsilon) := \log(K/\varepsilon) \), \( M := M(K, \varepsilon) = [K^2 e^{\nu(K, \varepsilon)}] \), \( F(K, a, b) := F(K, 0; a, b) \), \( p_a := \lfloor a \rfloor \), \( q_a := q_{a,b,K} = \lfloor a + 2bK \rfloor \), \( p_{1,a} := p_{1,a,b,K} = q_{a,b,K} - p_{a,b,K} \), \( \omega_a := \omega_{a,b,K} = \lfloor a + 2bK \rfloor \), and \( \omega_{1,a} := p_{a} - a \). Let \( \delta_n \) denote the function...
which is 1 for \( n = 0 \), and 0 otherwise, and let \( J(.) \) and \( \tilde{I}_C(.) \) be as defined in Section 2. Then for any tuple \( (\alpha, a, b) \in [-1, 1] \times [0, 2] \times [0, 1/4] \) such that \( p_{a+ax} < q_{a+ax} \) and \( a + \alpha x \in (0, 2) \) for all \( x \in [-1/4, 1/4] \), we have

\[
(6.22) \quad F(K; a + \alpha x, b) = e^{\pi i/4 - \pi i(a + \alpha x)^2/(2b)} F\left(\frac{2bK}{a + \alpha x}, -\frac{1}{4b}\right) + R_M(K, a + \alpha x, b) + O(K^{-2\varepsilon} e^{-K}),
\]

where \( x \) is any number in \([-1/4, 1/4]\), and \( R_M(K, a + \alpha x, b) \) is a linear combination of the functions \( J \), and the following eighteen functions:

\[
\begin{align*}
&J(K; M, 2bK - \omega_{a+ax}, b), \\
&J(K; p_{1, a+ax}, \omega_{a+ax}, b), \\
&J(K; M, 1 - \omega_{a+ax}, b), \\
&\tilde{I}_C(K; 1 - \omega_{a+ax}, b), \\
&\tilde{I}_C(K; \omega_{a+ax}, b), \\
&\frac{1}{\sqrt{2b}} e^{-\pi i(a+ax)^2/(2b)} e^{2\pi iax} K \tilde{I}_C(K; -i\omega_{a+ax} + 2bK, -ib), \\
&c_{1,a+ax} e^{2\pi iax/2b - \pi i(a+ax)^2/(2b)} e^{2\pi iax} K \tilde{I}_C(K; -i\omega_{a+ax} + 2bK, -ib), \\
&c_{2,a+ax} e^{2\pi iax} K \tilde{I}_C(K; -i\omega_{a+ax} + 2bK, -ib),
\end{align*}
\]

where \( c_{1,a} = \delta_{a} - p_{a} \), \( c_{2,a} := c_{2,a,b,K} = \delta_{q_{a,b,K} - K_{a,b,K}^* - 1} \), and \( c_{3,a} := c_{3,a,b,K} = \delta_{q_{a,b,K} - K_{a,b,K}^* - 2} \). The coefficients in the linear combination can all be computed to within \( \pm \varepsilon/K^2 \) say using \( O(\nu(K, \varepsilon)) \) operations on numbers of \( O(\nu(K, \varepsilon)) \) bits, are bounded by \( O(1) \), and do not depend on \( x \). Implicit asymptotic constants are absolute.

Proof. This follows directly from formulas (3.37), (3.38), and (3.39), the method of proof of Lemmas 6.2 and 6.3, the remarks following formulas (3.28) and (6.11), and some routine calculations and estimates. The conditions \( p_{a+ax} < q_{a+ax} \) and \( a + \alpha x \in (0, 2) \) for all \( x \in [-1/4, 1/4] \), which are stated in the lemma, are not essential but they help simplify the presentation of Lemma 6.7 next. □

Lemma 6.7. Let \( \varepsilon \in (0, e^{-1}) \), \( K > \Lambda(K, \varepsilon) := 1000 \nu(K, \varepsilon)^6 \) say, \( K \) an integer, and \( (\alpha, a, b) \in [-1/4, 1/4] \) such that \( p_{a+ax} < q_{a+ax} \) and \( a + \alpha x \in (0, 2) \) for all \( x \in [-1/4, 1/4] \). Let \( [w, z] \subset [-1/4, 1/4] \) be any subinterval such that \( p_{a+ax} \) and \( q_{a+ax} \) are constant over \( x \in [w, z] \), \( p_{a+ax} < q_{a+ax} \) for all \( x \in [w, z] \), and \( a + \alpha x \in (0, 2) \) for all \( x \in [w, z] \). Last, let \( l \) and \( m \) denote any integers satisfying \( m, l \in [0, 1000 \nu(K, \varepsilon)] \) say. Then for any \( x \in [w, z] \), each of the eighteen functions listed in Lemma 6.6 can be written as a linear combination of the functions

\[
x^m, \quad x^m \exp (2\pi iax K), \quad \exp \left(2\pi iax P/(2b) - 2\pi iax^2/(4b)\right),
\]
where \( P \in \{-1, 0, K^*, K^* + 1\} \), and the functions
\[
\exp \left( 2\pi i \omega_{a+\alpha x} N - 2\pi (1-i)m \frac{\omega_{a+\alpha x}}{\sqrt{2b}} \right) \times \int_0^1 t^l \exp \left( -2\pi (1-i) \frac{\omega_{a+\alpha x}}{\sqrt{2b}} t - 2\pi mt \right) dt,
\]
where \( N \in \{0, K\} \), and the functions
\[
(\omega_{a+\alpha x})^m \exp \left( 2\pi i \omega_{a+\alpha x} L - 2\pi \omega_{a+\alpha x} R \right),
\]
where \( L, R \in [K, K + 1000 \nu(K, \varepsilon)] \) say, as well as functions of the same form, but with \( \omega_{a+\alpha x} \) possibly replaced by \( 1 - \omega_{a+\alpha x} \) or \( \omega_{1,a+\alpha x} \) or \( 1 - \omega_{1,a+\alpha x} \), plus an error term bounded by \( O(\Lambda(K, \varepsilon)K^{-2}\varepsilon) \). The length of the linear combination is \( O(\nu(K, \varepsilon)) \) terms. The coefficients in the linear combinations can all be computed to within \( \pm \varepsilon/K^2 \) using \( O(\Lambda(K, \varepsilon)) \) operations on numbers of \( O(\nu(K, \varepsilon)^2) \) bits, are bounded by \( O(K) \), and are independent of \( x \). Implicit Big-O constants are absolute.

Proof. This follows from Lemma 6.6, the proofs of Lemmas 6.1 and 6.2, the assumption that \( p_{a+\alpha x} \) and \( q_{a+\alpha x} \) are constant over \( x \in [w, z] \), and some routine calculations.

Acknowledgment. I would like thank my Ph.D. thesis advisor Andrew Odlyzko. Without his help and comments this paper would not have been possible. I would like to thank Jonathan Bober, Dennis Hejhal, and Michael Rubinstein, for helpful remarks.

References


A METHOD TO COMPUTE THE TRUNCATED THETA FUNCTION


(Received: January 1, 2008)

Pure Mathematics, University of Waterloo, Waterloo, Ontario, Canada
E-mail: hiaryg@gmail.com