Computing Jacobi's θ in quasi-linear time

Hugo Labrande

Computing Jacobi's θ in quasi-linear time

Hugo Labrande[1,2,3,4]

[1] Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
[2] LORIA / INRIA Lorraine, CARAMEL team,
615 rue du jardin botanique, 54602 Villers-lès-Nancy Cedex, France
[3] CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
[4] University of Calgary, 2500 University Dr NW, Calgary, AB, Canada T2N1N4
hugo.labrande@inria.fr

Abstract

Jacobi's θ function has numerous applications in mathematics and computer science; a naive algorithm allows the computation of $\theta(z,\tau)$, for z,τ verifying certain conditions, with precision P in $O(\mathcal{M}(P)\sqrt{P})$ bit operations, where $\mathcal{M}(P)$ denotes the number of operations needed to multiply two complex P-bit numbers. We generalize an algorithm which computes specific values of the θ function (the *theta-constants*) in asymptotically faster time; this gives us an algorithm to compute $\theta(z,\tau)$ with precision P in $O(\mathcal{M}(P)\log P)$ bit operations, for any $\tau \in \mathcal{F}$ and z reduced using the quasi-periodicity of θ.

1 Introduction

Jacobi's θ function appears in a wide range of fields, such as non-linear differential equations (as a solution of the heat equation), the study of modular forms, and number theory, in which it is the main ingredient to convert between algebraic and analytic representations of elliptic curves. Namely, we have the embedding [12, I.4]

$$\begin{aligned} \mathbb{C}/(\mathbb{Z}+\tau\mathbb{Z}) &\to \mathbb{P}^3(\mathbb{C}) \\ z &\mapsto (\theta_{00}(2z,\tau),\theta_{01}(2z,\tau),\theta_{10}(2z,\tau),\theta_{11}(2z,\tau)) \end{aligned}$$

where the θ_i are essentially the θ function with its z argument translated. We also have the equation

$$\wp(z,\tau) = \frac{\pi^2}{3}(\theta_{10}^4(0,\tau)-\theta_{01}^4(0,\tau)) - \pi^2\theta_{01}^2(0,\tau)\theta_{10}^2(0,\tau)\frac{\theta_{00}^2(z,\tau)}{\theta_{11}^2(z,\tau)}$$

which allows to compute, for any point on the torus, its x-coordinate on the curve $E(\mathbb{C}) : y^2 = 4x^3 - g_2x - g_3$.

Special values of the θ function have interesting additional properties: the *theta-constants*, the value of θ at points $z = 0, \frac{1}{2}$ and $\frac{\tau}{2}$. As modular forms in τ, they are linked to other modular functions, such as the j-invariant or Dedekind's η function. Computing the value of the theta-constants allows one to compute the value of those modular functions; this has been used in record computations of class polynomials [7], which are interesting to generate safe cryptographic curves with the CM method.

The main problem we are dealing with here is to compute $\theta(z,\tau)$ with given absolute precision P, which allows us to compute the above embedding at any given precision. We will suppose throughout the article that (z,τ) satisfy certain conditions; the general case can be deduced from this one using formulas we mention later. The θ function is defined by a rapidly convergent series; under the conditions specified on z,τ, it gives a naive algorithm that requires a running time of $O(\mathcal{M}(P)\sqrt{P})$ bit operations, where $\mathcal{M}(P)$ is the number of operations needed to multiply two P-bit complex numbers. Although fast, this is a worse running time than other transcendental functions such as the exponential of a complex number, which can be computed in *quasi-optimal time* $O(\mathcal{M}(P)\log P)$.

There is an algorithm to compute the theta-constants asymptotically faster than with the naive method, outlined in [6]. This algorithm relies on connections between theta-constants and the *arithmetico-geometric mean* (AGM) of Gauss; the complex-valued AGM, when evaluated at the theta-constants, has interesting properties, and this is used along with Newton's method to, in a sense, invert the AGM and recover the values of the theta-constants. This algorithm allows computation of theta-constants for $\tau \in \mathcal{F}$ with precision P in quasi-optimal time $O(\mathcal{M}(P)\log P)$, independently of τ. It is faster than the naive method for precisions as low as a few thousand bits.

In this article, we provide a generalization of this algorithm which computes $\theta(z,\tau)$, for $\tau \in \mathcal{F}$ and z such that $\mathrm{Im}(z) \leq \mathrm{Im}(\tau)/2$, with absolute precision P in $O(\mathcal{M}(P)\log P)$ bit operations. We give two algorithms: the first one runs in quasi-optimal time in P, but its running time depends on z and τ; we then use this algorithm as a subroutine to build a quasi-optimal algorithm with complexity independent of z and τ, provided $\tau \in \mathcal{F}$ and $\mathrm{Im}(z) \leq \mathrm{Im}(\tau)/2$. An GNU MPC [8] implementation of the algorithm was realized; it is faster than the naive method for values of P greater than a few hundred thousand digits.

Our algorithm provides the six values $\theta(z,\tau), \theta(z+\frac{1}{2},\tau), \theta(z+\frac{\tau}{2},\tau), \theta(0,\tau), \theta(\frac{1}{2})$ and $\theta(\frac{\tau}{2})$, which are sufficient to compute the projective embedding mentioned above. It can also be used to compute the Weierstrass $\wp$ function and its derivative in quasi-optimal time; hence, this paper provides a quasi-optimal time algorithm to compute the "Jacobi map" $\mathbb{C}/\Lambda \to E(\mathbb{C})$ of an elliptic curve. We note that the "Abel map" $E(\mathbb{C}) \to \mathbb{C}/\Lambda$ can already be computed in quasi-optimal time using links to elliptic integrals and the Landen isogeny; see [2] and [5].

We note that [11] gives an algorithm to compute θ with real arguments (i.e. $\theta(u,m)$ with $0 < m < 1$ and $0 \leq u \leq K(m)$), defined by its representation as an infinite product, with the same, quasi-optimal complexity. Their algorithm relies on the Landen transform for the θ function, and could perhaps be generalized to the complex setting. We had independently pursued this line of thought, but found that in the complex setting, the presence of trigonometric functions induced heavy precision losses for some inputs; however, there may be a workaround those issues, which would allow one to find an algorithm for the complex setting and with quasi-optimal complexity relying on the Landen transform.

This article is organized as follows. We introduce the necessary mathematical background and the strategies we follow for argument reduction in Section 2, which justifies our choice to consider throughout the paper the case $\tau \in \mathcal{F}$ and $\mathrm{Im}(z) \leq \mathrm{Im}(\tau)/2$; we then provide an analysis of the naive algorithm for $\theta(z,\tau)$ under those conditions. Section 3 introduces a sequence derived from relations between values of θ, and we prove quadratic convergence of a certain homogeneization of the sequence; this is what replaces the AGM in the general case. Section 4 gives a first algorithm for computing θ-functions, with a complexity depending on z and τ; this is quasi-optimal provided that z,τ belong to a compact set. We then get rid of the dependency in z,τ much in the same way as in the case of theta-constants, which gives a uniform algorithm with complexity $O(\mathcal{M}(P)\log P)$. Section 5 shows timings for our GNU MPC implementation of this last algorithm and compare it to our implementation of the naive algorithm.

2 The function θ, and θ-constants

2.1 Definitions and argument reduction

We recall a few basic facts, following the presentation of [12].

Definition 2.1. *Define, for $z \in \mathbb{C}$ and $\tau \in \mathcal{H}$ (i.e. $\operatorname{Im}\tau > 0$)*

$$\theta(z,\tau) = \sum_{n\in\mathbb{Z}} \exp\left(\pi i\tau n^2 + 2\pi i n z\right).$$

Proposition 2.2 (quasi-periodicity)**.** *We have $\theta(z+1,\tau) = \theta(z,\tau)$ and $\theta(z+\tau,\tau) = e^{-\pi i z - 2\pi i\tau}\theta(z,\tau)$; in fact for any integers a, b,*

$$\theta(z + a\tau + b, \tau) = e^{-i\pi a^2\tau - 2i\pi a z}\theta(z,\tau). \tag{1}$$

We also define the following variants of the theta function (which are related to the definition of "theta functions with characteristics") [12, Section I.3]:

Definition 2.3.

$$\theta_{00}(z,\tau) = \theta(z,\tau) \qquad \theta_{10}(z,\tau) = \exp(\pi i\tau/4 + \pi i z)\,\theta\left(z + \frac{\tau}{2}, \tau\right)$$
$$\theta_{01}(z,\tau) = \theta\left(z + \frac{1}{2}, \tau\right) \qquad \theta_{11}(z,\tau) = \exp(\pi i\tau/4 + \pi i(z+1/2))\,\theta\left(z + \frac{\tau+1}{2}, \tau\right)$$

*We define **theta-constants** as the values in 0 of those functions.*

Those functions and their theta-constants are linked by a great number of formulas; we will give such formulas as we use them, and most of them can be found in [12, Section I.5]. Note that we have:

Proposition 2.4. *For any τ, the functions $z \mapsto \theta_{00}(z,\tau), z \mapsto \theta_{01}(z,\tau), z \mapsto \theta_{10}(z,\tau)$ are even, while $z \mapsto \theta_{11}(z,\tau)$ is odd.*

The latter implies that $\theta_{11}(0,\tau) = 0$, so the only theta constants we are interested in are $\theta_{00}(0,\tau), \theta_{01}(0,\tau)$, and $\theta_{10}(0,\tau)$.

Those properties can be used for the purpose of argument reduction. For instance, we can use the parity of θ to suppose that $\operatorname{Im}(z) \geq 0$; if this is not the case, one can consider $-z$ instead of z, which does not change the value of θ but ensures that $\operatorname{Im}(z) \geq 0$. Furthermore, Equation (1) can be used to recover $\theta(z,\tau)$ from the value of $\theta(z',\tau)$, where z' is such that $|\operatorname{Re}(z')| \leq \frac{1}{2}$ and $|\operatorname{Im}(z')| \leq \frac{\operatorname{Im}(\tau)}{2}$; the added cost is the cost of computing an exponential factor. This exponential factor can become quite big; should one want to compute $\theta(z,\tau)$ with an error of at most 2^{-P}, they have to work with representations of at least $P + C$ bits, with

$$C = \log_2(|\theta(z',\tau)|) + \pi\log_2(e)(a^2\operatorname{Im}(\tau) + 2a\operatorname{Im}(z)) + 2$$

This is because the integral part of the result fits in C bits, while the fractional part should be coded on at least P bits to ensure a final error bounded by 2^{-P}. The complexity of running our algorithm and computing the exponential factor will be $O(\mathcal{M}(P+C)\log(P+C))$, and hence will depend on τ and z; this is inevitable. Hence, throughout the paper we suppose that z is reduced, in the sense that $|\operatorname{Re}(z)| \leq \frac{1}{2}$ and $0 \leq \operatorname{Im}(z) \leq \frac{\operatorname{Im}(\tau)}{2}$, with the understanding that

the step of argument reduction has a complexity depending on the original values of z and τ. However, as Section 2.2 shows, this hypothesis, combined with a hypothesis in τ, allows us to work with values of θ bounded by 4, which allows us to write an algorithm with complexity only depending on P for any z satisfying these conditions.

We can also reduce the second argument of θ. Define the action of $SL_2(\mathbb{Z})$ on the complex upper-half plane $\mathcal{H}$ by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \tau \mapsto \frac{a\tau + b}{c\tau + d}.$$

Its fundamental domain is

$$\mathcal{F} = \{\omega \in \mathcal{H} \mid |\operatorname{Re}(\omega)| < 1/2, |\omega| > 1\}$$

Computing $\tau' \in \mathcal{F}$ and $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z})$ such that $\tau' = M\tau$ can be done by finding the shortest vector in the lattice $(1, \tau)$ using Gauss's algorithm [13], which (since the inertia is small) will be asymptotically negligible. The value of $\theta(z, \tau)$ can then be computed from $\theta(z', \tau')$ (for some value z') using the following theorem:

Theorem 2.5 (extension of [12, Theorem 7.1])**.** *Let $\tau \in \mathcal{H}$ and $z \in \mathbb{C}$, and let $\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z})$. Suppose $c > 0$, or $c = 0$ and $d > 0$; if not, take $-\gamma$. Then we have:*

$$\theta_i\left(\frac{z}{c\tau + d}, \frac{a\tau + b}{c\tau + d}\right) = \zeta_{i,\gamma,\tau}\sqrt{c\tau + d}\, e^{i\pi c z^2/(c\tau+d)}\theta_{\sigma(i)}(z, \tau) \tag{2}$$

where the square root is taken with positive real part, $\zeta_{i,\gamma,\tau}$ is an eighth root of unity and σ is a permutation of the elements $(00, 01, 10)$, defined by the following table:

a	b	c	d	$\sigma(00, 01, 10)$
odd	even	even	odd	$(00, 01, 10)$
odd	odd	even	odd	$(01, 00, 10)$
odd	even	odd	odd	$(10, 01, 00)$
even	odd	odd	even	$(00, 10, 01)$
odd	odd	odd	even	$(10, 00, 01)$
even	odd	odd	odd	$(01, 10, 00)$

Proof. Define for any $\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z})$:

$$e_\gamma(\tau) = c\tau + d, \qquad f_\gamma(z, \tau) = e^{i\pi c z^2/(c\tau+d)}$$

With a bit of care, one can prove

$$e_{\gamma_1}(\gamma_2\tau)e_{\gamma_2}(\tau) = e_{\gamma_1\gamma_2}(\tau), \qquad f_{\gamma_1}\left(\frac{z}{c_2\tau + d_2}, \gamma_2\tau\right) f_{\gamma_2}(z, \tau) = f_{\gamma_1\gamma_2}(z, \tau)$$

The maps $T : \tau \mapsto \tau + 1$ and $S : \tau \mapsto \frac{-1}{\tau}$ are generators of $SL_2(\mathbb{Z})$, and we have [12, Table V, p. 36]

$$\theta_i\left(z, T\tau\right) = \zeta_{i,T,\tau}\sqrt{e_T(\tau)} f_T(z, \tau)\theta_{\sigma_T(i)}(z, \tau), \qquad \theta_i\left(\frac{z}{\tau}, S\tau\right) = \zeta_{i,S,\tau}\sqrt{e_S(\tau)} f_S(z, \tau)\theta_{\sigma_S(i)}(z, \tau)$$

with square roots taken with real parts and for some $\zeta \in \mathbb{U}_8$ and $\sigma_S, \sigma_T \in \mathfrak{S}_3$. Hence for all $\gamma \in SL_2(\mathbb{Z})$,

$$\theta_i\left(\frac{z}{e_\gamma(\tau)^2}, \gamma\tau\right) = \zeta_{i,\gamma,\tau}\sqrt{e_\gamma(\tau)}f_\gamma(z,\tau)\theta_{\sigma_\gamma(i)}(z,\tau)$$

for some root of unity and some permutation. The correspondance $\gamma \mapsto \sigma_\gamma$ can be determined from [12, p. 36], although one can simply notice it is independent of z and use the tables found by Gauss in the case of theta-constants [4, Eq. 2.15]. Finally, we could attempt to give a formula for $\zeta_{i,\gamma,\tau}$, but it is more efficient to simply compute a very low precision approximation of $\theta(z,\tau)$ and compare it to the full-precision value to determine which eighth root is needed.

□

Thus, in order to recover $\theta_i(z,\tau)$ from $\theta_i\left(\frac{z}{c\tau+d}, \frac{a\tau+b}{c\tau+d}\right)$, one needs to compute $\sqrt{c\tau+d}$ (which is done in $O(\mathcal{M}(P))$ bit operations) and $e^{\pi i c z^2/(c\tau+d)}$ (done in $O(\mathcal{M}(P)\log P)$ bit operations), and perform a division; determining ζ is asymptotically negligible. The cost of this step is then $O(\mathcal{M}(P)\log P)$ bit operations.

We note that in general, because of the permutation σ_γ, we need to have computed all three values $\theta_{00,01,10}\left(\frac{z}{c\tau+d}, \frac{a\tau+b}{c\tau+d}\right)$ in order to be able to use the formula to compute, say, $\theta_{00}(z,\tau)$. We will occasionally talk about computing θ_{11}, but this will not be the focus of the paper. Hence, the problem we consider in this paper is the following:

$$\begin{aligned}
&\text{Compute} && \theta_{00}(z,\tau), \theta_{00}(0,\tau), \theta_{01}(z,\tau), \theta_{01}(0,\tau), \theta_{10}(z,\tau), \theta_{10}(0,\tau) \\
&\text{where} && |\tau| > 1, \quad |\operatorname{Re}(\tau)| \le \frac{1}{2}, \quad \operatorname{Im}(\tau) > 0, \\
& && |\operatorname{Re}(z)| \le \frac{1}{2}, \quad 0 \le \operatorname{Im}(z) \le \frac{\operatorname{Im}(\tau)}{2} \qquad (3) \\
&\text{in quasi-optimal time,} && \text{i.e. } O(\mathcal{M}(P)\log P).
\end{aligned}$$

2.2 Naive algorithm to compute θ

2.2.1 Partial summation of the series defining θ

We define the following partial summation for the series defining $\theta(z,\tau)$:

$$S_B(z,\tau) = 1 + \sum_{0<n<B} q^{n^2}(e^{2i\pi nz} + e^{-2i\pi nz})$$

where use the notation $q = e^{i\pi\tau}$. We have

Proposition 2.6. *Suppose that* $\operatorname{Im}(\tau) \ge 0.35$ *and* $0 \le \operatorname{Im}(z) \le \operatorname{Im}(\tau)/2$*; in particular, this is the case if the conditions* (3) *are satisfied. Then, for* $B \ge 1$, $|\theta(z,\tau) - S_B(z,\tau)| \le 3|q|^{(B-1)^2}$.

Proof. We look at the remainder of the series:

$$\begin{aligned}
|\theta(z,\tau) - S_B(z,\tau)| &\leq \sum_{n\geq B} |q|^{n^2}(|e^{2i\pi nz}| + |e^{-2i\pi nz}|) \\
&\leq \sum_{n\geq B} |q|^{n^2}(1+|q|^{-n}) \leq 2\sum_{n\geq B} |q|^{n^2-n} \\
&\leq 2\sum_{n\geq B} |q|^{(n-1)^2} \leq 2\sum_{n\geq 0} |q|^{(B-1+n)^2} \\
&\leq 2|q|^{(B-1)^2} \sum_{n\geq 0} |q|^{2n(B-1)+n^2} \\
&\leq 2\frac{|q|^{(B-1)^2}}{1-|q|^{2B-1}} \qquad (4)
\end{aligned}$$

A numerical calculation shows that for $\mathrm{Im}(\tau) \geq 0.35$, we have $\frac{2}{1-|q|} \leq 3$, which proves the proposition. □

We can prove the same inequality for θ_{01}, since the series that define it has the same terms, up to sign, as the series for θ. Note that, unlike the analysis of [6] for naive theta-constant evaluation, we cannot get a bound for the relative precision: since $\theta(\frac{1+\tau}{2}, \tau) = 0$, there is no lower bound for $|\theta(z,\tau)|$.[1] If we set

$$B(P,\tau) = \left\lceil \sqrt{\frac{P+2}{\pi\,\mathrm{Im}(\tau)\log_2(e)}} \right\rceil + 1.$$

we have $4|q|^{(B-1)^2} \leq 2^{-P}$, which means the approximation is accurate with absolute precision P. We just showed that:

Theorem 2.7. *To compute $\theta(z,\tau)$ with absolute precision P bits, it is enough to sum over all $k \in \mathbb{Z}$ such that*

$$|k| \leq \left\lceil \sqrt{\frac{P+2}{\pi\,\mathrm{Im}(\tau)\log_2(e)}} \right\rceil + 1$$

Note that this bound is larger than the one of [6, p. 5].

2.2.2 Naive algorithm

We then present a naive algorithm to compute not only the value of $\theta(z,\tau)$, but also the value of $\theta_{01}(z,\tau), \theta_{00}(0,\tau), \theta_{01}(0,\tau)$ for only a marginal amount of extra computation; this is the algorithm we will use for comparison to the fast algorithm we propose in this article. The algorithm performs computations at a precision $\mathcal{P}$, which we determine later so that the result is accurate to the desired precision P.

Define the sequence $(v_n)_{n\in\mathbb{N}}$ as

$$v_n = q^{n^2}(e^{2i\pi nz} + e^{-2i\pi nz})$$

so that $\theta(z,\tau) = 1 + \sum_{n\geq 1} v_n$. This satisfies the following recurrence relation for $n > 1$:

$$v_{n+1} = q^{2n} v_1 v_n - q^{4n} v_{n-1}.$$

[1] Incidentally, this is why we consider only absolute precision in this paper.

We use this recursion formula to compute v_n efficiently, which is similar to the trick used by [9, Prop. 3]. This removes the need for divisions and the need to compute and store $e^{-2i\pi nz}$, which can get quite big; indeed, computing it only to multiply it by the very small q^{n^2} is wasteful. The resulting algorithm is Algorithm 1.

Algorithm 1 Compute $\theta_{00,01}(z,\tau), \theta_{00,01}(0,\tau)$ for z,τ satisfying conditions (3).

prec $\leftarrow \mathcal{P}$
$B \leftarrow \left\lceil \sqrt{\frac{P+2}{\pi \operatorname{Im}(\tau) \log_2(e)}} \right\rceil + 1$
$\theta_{0,z} \leftarrow 1, \theta_{1,z} \leftarrow 1, \theta_{0,0} \leftarrow 1, \theta_{1,0} \leftarrow 1$
$q \leftarrow e^{i\pi\tau}, \quad q_1 \leftarrow q, \quad q_2 \leftarrow q$
$v_1 \leftarrow e^{2i\pi(z+\tau/2)} + e^{-2i\pi(z-\tau/2)}, \quad v \leftarrow v_1, \quad v' \leftarrow 2$
for $n = 1..B$ **do**
    /* $q_1 = q^n, q_2 = q^{n^2}, v = v_n, v' = v_{n-1}$ */
    $\theta_{0,z} \leftarrow \theta_{0,z} + v, \quad \theta_{1,z} \leftarrow \theta_{1,z} + (-1)^i \times v$
    $\theta_{0,0} \leftarrow \theta_{0,0} + 2q_2, \quad \theta_{1,0} \leftarrow \theta_{1,0} + (-1)^i \times 2q_2$
    $q_2 \leftarrow q_2 \times (q_1)^2 \times q$
    $q_1 \leftarrow q_1 \times q$
    temp $\leftarrow v, \quad v \leftarrow \left(q_1^2 \times v_1\right) \times v - q_1^4 \times v' \quad v' \leftarrow$ temp
end for

2.2.3 Error analysis and complexity

We have the following theorem:

Theorem 2.8. *For z,τ satisfying conditions* (3)*, Algorithm 1 with $\mathcal{P} = P + \log B + 7$ computes $\theta_{00}(z,\tau)$, $\theta_{01}(z,\tau)$, $\theta_{00}(0,\tau)$, $\theta_{01}(0,\tau)$ with absolute precision P bits. This gives an algorithm which has bit complexity $O\left(\mathcal{M}(P)\sqrt{\frac{P}{\operatorname{Im}(\tau)}}\right)$.*

Performing the analysis of this algorithm requires bounding the error that is incurred during the computation. We then compensate the number of inaccurate bits by increasing the precision. We use the following theorem:

Theorem 2.9. *For $j = 1, 2$, let $z_j = x_j + iy_j \in \mathbb{C}$ and $\tilde{z_j} = \tilde{x_j} + i\tilde{y_j}$ its approximation. Suppose that $|z_j - \tilde{z_j}| \leq k_j 2^{-P}$ and that $k_j \leq 2^{P/2}$. Then*

1. $|\operatorname{Re}(z_1 + z_2) - \operatorname{Re}(\tilde{z_1} + \tilde{z_2})| \leq (k_1 + k_2)2^{-P}$
2. $|\operatorname{Re}(z_1 z_2) - \operatorname{Re}(\tilde{z_1}\tilde{z_2})| \leq (2 + 2k_1|z_2| + 2k_2|z_1|)2^{-P}$
3. $|\operatorname{Re}(z_1^2) - \operatorname{Re}(\tilde{z_1}^2)| \leq (2 + 4k_1|z_1|)2^{-P}$

and the same bounds apply to imaginary parts as well; and

4. $|e^{z_1} - e^{\tilde{z_1}}| \leq |e^{z_1}| \frac{7k_1+8.5}{2} 2^{-P}$.

Furthermore if $|z_j| \geq 2k_j 2^{-P}$,

5. $|\operatorname{Re}\left(\frac{z_1}{z_2}\right) - \operatorname{Re}\left(\frac{\tilde{z_1}}{\tilde{z_2}}\right)| \leq \left(\frac{6(2+2k_1|z_2|+2k_2|z_1|)}{|z_2|^2} + \frac{2(4+8k_2|z_2|)(2|z_1||z_2|+1)+2}{|z_2|^4}\right) 2^{-P}$

and the same bound applies to the imaginary part, and

6. $|\sqrt{z_1} - \sqrt{\tilde{z_1}}| \leq \frac{k_1}{\sqrt{|z_1|}} 2^{-P}$.

This theorem is not very hard to prove; we refer to [10] for details.

Proof of Theorem 2.8. We first determine the size of the quantities we are manipulating; this is needed to evaluate the error incurred during the computation, as well as the number of bits needed to store fixed-point approximations of absolute precision P of the intermediate quantities. Taking $B = 1$ in Proposition 2.6 gives $|\theta(z,\tau) - 1| \leq 3$, so $|\theta(z,\tau)| \leq 4$; actually, this also proves $|S_B(z,\tau)| \leq 4$, which means that $|\theta_{0,z}|, |\theta_{1,z}|, |\theta_{0,0}|, |\theta_{1,0}|$ are bounded by 4. We also have $|q| \leq 0.07$, and $|q_2| \leq |q|^{n^2} \leq |q|^n = |q_1| \leq |q| \leq 0.07$. As for the v_i, we have $v_0 = 2$, and for $n \geq 1$

$$|v_n| \leq |q|^{n^2+n} + |q|^{n^2-n} \leq (1 + |q|^{2n}) q^{n^2-n} \leq 1.0049 q^{n^2-n} \leq 1.0049$$

Hence, storing all the complex numbers above, including our result, with absolute precision P only requires $P + 2$ bits, since their integral part is coded on only 2 bits. Note that, had we computed $e^{-2i\pi nz}$ before multiplying it by q^{n^2}, we would have needed $O(\mathrm{Im}(\tau))$ more bits, which worsens the asymptotic complexity.

Computing the absolute precision lost during this computation is done using Theorem 2.9. We start with the bounds $|\tau - \tilde{\tau}| \leq \frac{1}{2} 2^{-\mathcal{P}}$ and $|z - \tilde{z}| \leq \frac{1}{2} 2^{-\mathcal{P}}$, coming from the hypothesis that the approximations of z and τ are correctly rounded with precision $\mathcal{P}$. We then need to estimate k_{v_1} and k_q, which can be done using the formula giving the absolute error when computing an exponential from Theorem 2.9. Given that $\tau \in \mathcal{F}$, we have

$$\begin{aligned} |q - \tilde{q}| &\leq 0.07 \frac{7 \times 1/2 + 8.5}{2} 2^{-P} \leq 0.42 \times 2^{-P} \\ |v_1 - \tilde{v_1}| &\leq 6(|e^{-\pi(\mathrm{Im}(\tau) + 2\,\mathrm{Im}(z))}| + |e^{\pi(2\,\mathrm{Im}(z) - \mathrm{Im}(\tau))}|) \times 2^{-P} \\ &\leq 6(|q| + 1) 2^{-P} \leq 6.42 \times 2^{-P} \end{aligned}$$

which means that $k_q \leq 0.42$ and $k_{v_1} \leq 6.42$. We then need to evaluate the loss of precision for each variable and at each step of the algorithm, which gives recurrence relations with non-constant coefficients. Solving those is rather tedious, and we use loose upper bounds to simplify the computation; we do not detail this proof in the present article. The results obtained by this method show that the error on the computation of the theta-constants is bounded by $(0.3B + 105.958)2^{-\mathcal{P}}$, and the one on the computation of the theta function is smaller than $(5.894B + 28.062)2^{-\mathcal{P}}$. This proves that the number of bits lost is bounded by $\log_2 B + c$, where c is a constant smaller than 7; hence we set $\mathcal{P} = P + \log B + 7$.

Finally, evaluating π and $\exp(z)$ with precision $\mathcal{P}$ can be done in $O(\mathcal{M}(\mathcal{P}) \log \mathcal{P})$ [1], but this is negligible asymptotically. In the end, computing an approximation up to 2^{-P} of $\theta(z,\tau)$ can be done in $O\left(\mathcal{M}\left(P + \log(P/\,\mathrm{Im}(\tau)) + c\right) \sqrt{\frac{P}{\mathrm{Im}(\tau)}}\right) = O\left(\mathcal{M}(P) \sqrt{\frac{P}{\mathrm{Im}(\tau)}}\right)$ bit operations. □

2.2.4 Computing θ_{10}

We mentioned in section 2.1 the need to compute $\theta_{10}(z,\tau)$ and $\theta_{10}(0,\tau)$ as well. One could think of recovering those values using Jacobi's quartic formula and the equation of the variety:

$$\theta_{00}(0,\tau)^4 = \theta_{01}(0,\tau)^4 + \theta_{10}(0,\tau)^4 \tag{5}$$

$$\theta_{00}^2(z,\tau)\theta_{00}^2(0,\tau) = \theta_{01}^2(z,\tau)\theta_{01}^2(0,\tau) + \theta_{10}^2(z,\tau)\theta_{10}^2(z,\tau) \tag{6}$$

that is to say, compute

$$\begin{aligned}\theta_{10}(0,\tau) &= \left(\theta_{00}(0,\tau)^4 - \theta_{01}(0,\tau)^4\right)^{1/4} \\ \theta_{10}(z,\tau) &= \frac{\sqrt{\theta_{00}^2(z,\tau)\theta_{00}^2(0,\tau) - \theta_{01}^2(z,\tau)\theta_{01}^2(0,\tau)}}{\theta_{10}(0,\tau)}.\end{aligned}$$

However, this approach induces an asymptotically large loss of absolute precision for both $\theta_{10}(0,\tau)$ and $\theta_{10}(z,\tau)$. According to Theorem 2.9, both square root extraction and inversion induce a loss of precision proportional to $|z|^{-1}$; since $\theta_{10}(0,\tau) \sim 4q^{1/2}$, the number of bits lost by applying those formulas is $O(\mathrm{Im}(\tau))$. Note that those formulas would also induce a big loss in relative precision; since $\theta_{00}(0,\tau)$ and $\theta_{01}(0,\tau)$ are very close when $\mathrm{Im}(\tau)$ goes to infinity, the subtraction induces a relative precision loss of $O(\mathrm{Im}(\tau))$ bits (for more details, see [6, Section 6.3]). Either of those analyses show that, in order to compensate precision loss, the naive algorithm should actually be run with a precision of $O(P + \log B + \mathrm{Im}(\tau))$, which gives a running time that worsens, insteads of getting better, when $\mathrm{Im}(\tau)$ gets big. We do not recommend this approach.

Instead, one should compute partial summations of the series defining θ_{10}, much in the same way as we did for $\theta(z,\tau)$. We outline the analysis in this case, which is very similar to the one for θ: supposing $n \geq 2$, we have $n^2 - 2n \geq (n-2)^2$, which can be used to prove that $|\theta_{10}(z,\tau) - S_B| \leq 3|q|^{(B-2)^2}$, so that the bound on B is thus just one more than for θ; the recurrence relation is the same; $q^{2n}|v_1|$ is bounded by 2 instead of 1, which in the worst case means $\log B$ more guard bits are needed. In what follows, we will refer to this algorithm as "the naive algorithm to compute $\theta_{10}(0,\tau), \theta_{10}(z,\tau)$"; its asymptotic complexity is, just like Algorithm 1, $O\left(\mathcal{M}(P)\sqrt{\frac{P}{\mathrm{Im}(\tau)}}\right)$ bit operations, which gets better as $\mathrm{Im}(\tau)$ increases.

We note that similar considerations apply to the problem of computing θ_{11}. One can compute $\theta_{11}(z,\tau)$ using the formula [12, p.22]

$$\theta_{11}(z,\tau)^2 = \frac{\theta_{01}(z,\tau)^2\theta_{10}(0,\tau)^2 - \theta_{10}(z,\tau)^2\theta_{01}(0,\tau)^2}{\theta_{00}(0,\tau)^2}. \tag{7}$$

Using this formula loses only a few bits of precision since $\theta_{00}(0,\tau)$ is bounded; however, one then needs to compute a square root, which potentially loses $O(\mathrm{Im}(\tau))$ bits. Hence, a summation of the series, which directly gives θ_{11}, is preferable.

2.3 Fast computation of theta-constants

Recall the definition of the arithmetico-geometric mean (AGM) for two positive real numbers a, b:

$$\begin{aligned} a_0 &= a, & b_0 &= b \\ a_{n+1} &= \frac{a_n + b_n}{2} & b_{n+1} &= \sqrt{a_n b_n} \end{aligned} \tag{8}$$

The sequences $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ both converge to the same limit, called the **arithmetico-geometric mean** of a and b. Furthermore, (a_n) and (b_n) are quadratically convergent, in the sense of the following definition:

Definition 2.10. *A sequence (a_n) is said to be **quadratically convergent** (to a limit ℓ) if there is a $C > 0$ such that for n large enough:*

$$|a_{n+1} - a_n| \leq C|a_n - a_{n-1}|^2$$

The constant C in the case of the AGM can be taken as $\frac{\pi}{8\min(|a|,|b|)}$ [6, Thm. 1]. Quadratic convergence implies that the number of exact digits approximately doubles with each iteration, so that one only needs $O(\log P)$ iterations to compute $\mathrm{AGM}(a,b)$ with precision P; hence the total cost of computing $\mathrm{AGM}(a,b)$ is $O(\mathcal{M}(P)\log P)$ bit operations.

It is possible to generalize the AGM to complex numbers, but there are two possibilities for the choice of the square root at each step. We then call **an AGM sequence for a and b** any sequence $(a_n,b_n)_{n\in\mathbb{N}}$ such that

$$a_0=a,\quad b_0=b,\qquad 2a_{n+1}=a_n+b_n,\quad b_{n+1}^2=a_nb_n$$

Note that there are uncountably many AGM sequences for a,b. We define unambiguously *the* AGM of two complex numbers following [4]:

Proposition 2.11. *Let $a,b\in\mathbb{C}$ and let $(a_n,b_n)_{n\in\mathbb{N}}$ be an AGM sequence for a and b. We say that the choice of signs is **good** at the rank n if*

$$|a_n-b_n|<|a_n+b_n|\qquad or\quad |a_n-b_n|=|a_n+b_n| \text{ and } \mathrm{Im}\left(\frac{b_n}{a_n}\right)>0$$

*We call the AGM sequence for a and b in which all the choices of signs are good the **optimal AGM sequence**, and define $\mathrm{AGM}(a,b)$ as the limit of the optimal AGM sequence for a and b.*

Finally we have the following proposition:

Proposition 2.12 ([4, Proposition 2.1])**.** *Let $(a_n,b_n)_{n\in\mathbb{N}}$ be an AGM sequence for a,b:*

- *If (a_n,b_n) has infinitely many bad choices of sign, $\lim_{n\to\infty}a_n=0$ and the convergence is at least linear;*
- *If (a_n,b_n) has only finitely many bad choices of sign (for instance if it is optimal), (a_n) and (b_n) both converge quadratically to the same non-zero limit.*

The link between the complex AGM and theta-constants is well-known:

Proposition 2.13. *We have the following formulas linking theta-constants:*

$$\theta_{00}^2(0,2\tau) = \frac{\theta_{00}^2(0,\tau)+\theta_{01}^2(0,\tau)}{2} \tag{9}$$

$$\theta_{01}^2(0,2\tau) = \theta_{00}(0,\tau)\theta_{01}(0,\tau) \tag{10}$$

This shows that $(\theta_{00}^2(0,2^n\tau),\theta_{01}(0,2^n\tau))_{n\in\mathbb{N}}$ is an AGM sequence for $\theta_{00}^2(0,\tau)$ and $\theta_{01}^2(0,\tau)$, and it converges quadratically to 1. Whether or not this sequence is the optimal AGM sequence is controlled by the following result:

Proposition 2.14 ([6, Theorem 2] or [4, Lemma 2.9])**.** *Define*

$$\mathcal{F}_{k'}=\left\{\tau\in\mathcal{H} \text{ such that } |\mathrm{Re}(\tau)|<1,\left|\tau+\frac{3}{4}\right|\geq\frac{1}{4},\left|\tau+\frac{1}{4}\right|>\frac{1}{4},\left|\tau-\frac{1}{4}\right|\geq\frac{1}{4},\left|\tau-\frac{3}{4}\right|>\frac{1}{4}\right\}\subset\mathcal{H}$$

Let $\tau\in\mathcal{F}_{k'}$, and let $(a_n,b_n)_{n\in\mathbb{N}}$ the optimal AGM sequence for $\theta_{00}^2(0,\tau)$ and $\theta_{01}^2(0,\tau)$. Then for all n we have $(a_n,b_n)=(\theta_{00}^2(0,2^n\tau),\theta_{01}^2(0,2^n\tau))$, which implies $\mathrm{AGM}(\theta_{00}^2(0,\tau),\theta_{01}^2(0,\tau))=1$.

In [6, Algorithm 4], an algorithm relying on the AGM, and with complexity $O(\mathcal{M}(P)\log P)$, is given to compute the value of the theta-constants with precision P bits. The algorithm uses the fact that $k'(\tau) = \frac{\theta_{01}^2(0,\tau)}{\theta_{00}^2(0,\tau)}$ is a solution to the following equation:

$$i\,\mathrm{AGM}(1,z) - \tau\,\mathrm{AGM}\left(1,\sqrt{1-z^2}\right) = 0$$

which is a consequence of the action of $SL_2(\mathbb{Z})$ on the theta-constants, as well as of Jacobi's quartic formula (Equation (5)). Newton's method, when given an approximation of $k'(\tau)$ with precision $P/2$ as well as the knowledge of τ with precision P, computes an approximation of $k'(\tau)$ with precision $P-\delta$, where δ is a small constant. If one carries out Newton's method while doubling the working precision with each iteration, it is asymptotically only as costly as the last iteration; this means $k'(\tau)$ can be computed with precision P in quasi-optimal running time. One can then recover the individual values of the theta-constants using the equation

$$\mathrm{AGM}\left(1,\frac{\theta_{01}(0,\tau)^2}{\theta_{00}(0,\tau)^2}\right) = \frac{1}{\theta_{00}(0,\tau)^2}. \tag{11}$$

However, the complexity of this algorithm is not uniform – that is to say, it reaches this complexity only for τ within a compact set. A variant of the algorithm is proposed in [6, Algorithm 5] which makes the complexity uniform: if $P \leq 2\log \mathrm{Im}(\tau)$, use the naive algorithm (which gives the right complexity); if not, compute the value of the theta-constants at $\frac{\tau}{2^n}$ for some $n \leq \log \mathrm{Im}(\tau)$, and use the AGM to compute the theta-constants at τ. This gives an algorithm which complexity does not depend on τ.

3 A sequence related to θ-functions

3.1 Definition of the F sequence

We start with the following formula:

Proposition 3.1.

$$\theta_{00}(z,2\tau)^2 = \frac{\theta_{00}(z,\tau)\theta_{00}(0,\tau) + \theta_{01}(z,\tau)\theta_{01}(0,\tau)}{2} \tag{12}$$

$$\theta_{01}(z,2\tau)^2 = \frac{\theta_{00}(z,\tau)\theta_{01}(0,\tau) + \theta_{01}(z,\tau)\theta_{00}(0,\tau)}{2} \tag{13}$$

This formula is called in [3, formula 3.13, p.39] the change of basis formula from the $\mathcal{F}_2$ basis to the $\mathcal{F}_{(n,2)^2}$ basis. However a direct proof can be obtained with limited effort, using the series defining θ and some manipulations and term reorganization akin to

$$\sum_{n+m\equiv 0 \pmod 2} q^{n^2+m^2} = \sum_{i,j\in\mathbb{Z}} q^{(i+j)^2+(i-j)^2}.$$

We also note that one can similarly prove the following formula, which will be used in section 4.2:

$$\theta_{10}(z,2\tau)^2 = \frac{\theta_{00}(z,\tau)\theta_{00}(0,\tau) - \theta_{01}(z,\tau)\theta_{01}(0,\tau)}{2} \tag{14}$$

We then define the following function:

$$\begin{aligned} F : \mathbb{C}^4 &\to \mathbb{C}^4 \\ (x,y,z,t) &\mapsto \left(\frac{\sqrt{x}\sqrt{z}+\sqrt{y}\sqrt{t}}{2}, \frac{\sqrt{x}\sqrt{t}+\sqrt{y}\sqrt{z}}{2}, \frac{z+t}{2}, \sqrt{z}\sqrt{t}\right) \end{aligned}$$

Hence, according to Proposition 2.13 and 3.1, for some appropriate choice of roots we have

$$F\left(\theta_{00}^2(z,\tau),\theta_{01}^2(z,\tau),\theta_{00}^2(0,\tau),\theta_{01}^2(0,\tau)\right)=\left(\theta_{00}^2(z,2\tau),\theta_{01}^2(z,2\tau),\theta_{00}^2(0,2\tau),\theta_{01}^2(0,2\tau)\right).$$

Remark. One can also write rewrite F using Karatsuba-like techniques

$$F(x,y,z,t)=\left(\begin{array}{l}\dfrac{(\sqrt{x}+\sqrt{y})(\sqrt{z}+\sqrt{t})+(\sqrt{x}-\sqrt{y})(\sqrt{z}-\sqrt{t})}{4},\\ \dfrac{(\sqrt{x}+\sqrt{y})(\sqrt{z}+\sqrt{t})-(\sqrt{x}-\sqrt{y})(\sqrt{z}-\sqrt{t})}{4},\\ \dfrac{(\sqrt{z}+\sqrt{t})^2+(\sqrt{z}-\sqrt{t})^2}{4},\dfrac{(\sqrt{z}+\sqrt{t})^2-(\sqrt{z}-\sqrt{t})^2}{4}\end{array}\right) \tag{15}$$

to speed up computations. □

Following section 2.3, we define a **good choice** for square roots at the rank n as the following conditions being satisfied:

- $\mathrm{Re}(\sqrt{x_n})\geq 0, \qquad \mathrm{Re}(\sqrt{z_n})\geq 0;$
- $|\sqrt{x_n}-\sqrt{y_n}|<|\sqrt{x_n}+\sqrt{y_n}|, \quad$ or $|\sqrt{x_n}-\sqrt{y_n}|=|\sqrt{x_n}+\sqrt{y_n}|$ and $\mathrm{Im}\left(\frac{\sqrt{y_n}}{\sqrt{x_n}}\right)>0;$
- $|\sqrt{z_n}-\sqrt{t_n}|<|\sqrt{z_n}+\sqrt{t_n}|, \qquad$ or $|\sqrt{z_n}-\sqrt{t_n}|=|\sqrt{z_n}+\sqrt{t_n}|$ and $\mathrm{Im}\left(\frac{\sqrt{t_n}}{\sqrt{z_n}}\right)>0.$

The last condition is equivalent to $|z_n-t_n|\leq|z_n+t_n|$, which means that (z_n,t_n) is an AGM sequence for (z_0,t_0) in which all the choices of sign are good. Note that the condition $|x-y|<|x+y|$ is equivalent to $\mathrm{Re}\left(\frac{y}{x}\right)>0$.

Again, similarly to the AGM, for any $x,y,z,t\in\mathbb{C}$ we define the **optimal F sequence** $((x_n,y_n,z_n,t_n))_{n\in\mathbb{N}}$ as follows:

$$\begin{array}{rcl}(x_0,y_0,z_0,t_0) & = & (x,y,z,t)\\ (x_{n+1},y_{n+1},z_{n+1},t_{n+1}) & = & F\left(x_n,y_n,z_n,t_n\right)\end{array}$$

where all the choices of sign for the square roots are good. The study of this sequence and its convergence is done in Section 3.4.

3.2 Link with θ-functions

3.2.1 More argument reduction

We go slightly further than the conditions (3) in order to justify the forthcoming results. We wish to further reduce z, as follows:

$$0\leq\mathrm{Im}(z)\leq\frac{\mathrm{Im}(\tau)}{4}, \qquad |\,\mathrm{Re}(z)|\leq\frac{1}{8} \tag{16}$$

The first hypothesis allows us to avoid $z=\frac{\tau+1}{2}$, which is a zero of $\theta(z,\tau)$, and hence a pole of quotients of the form $\frac{\theta_i}{\theta_{00}}$, which we consider in our algorithm much in the same way as [6]. We prove Lemma 3.3 and Theorem 3.4 under this assumption. The second condition complements the first one as follows:

Lemma 3.2. *Let z, τ such that $|\operatorname{Re}(\tau)| \leq \frac{1}{2}$ and the conditions of* (16) *are verified. Then $\frac{z}{\tau}, \frac{-1}{\tau}$ verify the first condition of* (16)*.*

Proof. Write

$$\left|\operatorname{Im}\left(\frac{z}{\tau}\right)\right| = \frac{1}{|\tau|^2}|\operatorname{Im}(z)\operatorname{Re}(\tau) - \operatorname{Re}(z)\operatorname{Im}(\tau)| \leq \frac{\operatorname{Im}(\tau)}{|\tau|^2}\left(\frac{1}{4}\frac{1}{2} + \frac{1}{8}\right) = \frac{1}{4}\operatorname{Im}\left(\frac{-1}{\tau}\right).$$

□

This will be used to apply Theorem 3.4 to $\frac{z}{\tau}, \frac{-1}{\tau}$ in Proposition 4.1.

Note that those conditions are satisfied if one takes $z' = \frac{z}{2}$ with z satisfying conditions (3). One can then compute $\theta^2_{00,01}(z,\tau)$ from $\theta^2_{00,01,10}(z/2,\tau)$ and $\theta^2_{00,01,10}(0,\tau)$ using the following *z-duplication formulas* [12, p. 22]:

$$\begin{aligned}
\theta_{00}(z,\tau)\theta^3_{00}(0,\tau) &= \theta^4_{01}(z/2,\tau) + \theta^4_{10}(z/2,\tau) \\
\theta_{01}(z,\tau)\theta^3_{01}(0,\tau) &= \theta^4_{00}(z/2,\tau) - \theta^4_{10}(z/2,\tau) \\
\theta_{10}(z,\tau)\theta^3_{10}(0,\tau) &= \theta^4_{00}(z/2,\tau) - \theta^4_{01}(z/2,\tau).
\end{aligned} \tag{17}$$

This requires the knowledge of $\theta_{10}(z,\tau)$ and the associated theta-constant; this could be computed using Jacobi's formula (Equation (5)) and the equation of the variety (Equation (6)), but we end up using a different trick in our final algorithm.

3.2.2 Good choices of sign and thetas

We now prove that, for the arguments we consider, the good choices of sign correspond exactly to values of θ:

Lemma 3.3. *For any τ such that $\operatorname{Im}(\tau) \geq 0.345$ (in particular, for $\tau \in \mathcal{F}$) and z verifying the first condition in* (16) *we have*

$$|\theta_{00}(z,\tau) - \theta_{01}(z,\tau)| < |\theta_{00}(z,\tau) + \theta_{01}(z,\tau)|,$$

which also proves that $\operatorname{Re}\left(\frac{\theta_{01}(z,\tau)}{\theta_{00}(z,\tau)}\right) > 0, \operatorname{Re}\left(\frac{\theta_{01}(0,\tau)}{\theta_{00}(0,\tau)}\right) > 0.$

Proof. Write:

$$\begin{aligned}
|\theta_{00}(z,\tau) + \theta_{01}(z,\tau) - 2| &\leq 2\sum_{n\geq 2, n \text{ even}} |q^{n^2}(w^{2n} + w^{-2n})| \\
&\leq 2\sum_{n\geq 2, n \text{ even}} |q|^{n^2}(1 + |q|^{-n/2}) \\
&\leq 2\sum_{n\geq 1} |q|^{4n^2}(1 + |q|^{-n}) \\
&\leq 2|q|^3 + 2|q|^4 + \frac{2q^{16}}{1-q^{20}} + \frac{2q^{14}}{1-q^{19}} \\
|\theta_{00}(z,\tau) - \theta_{01}(z,\tau)| &\leq 2\sum_{n\geq 1, n \text{ odd}} |q|^{n^2}(1 + |q|^{-n/2}) \\
&\leq 2|q|^{1/2} + 2|q| + \frac{2q^9}{1-q^{16}} + \frac{2q^{7.5}}{1-q^{19}}
\end{aligned}$$

We have

$$\frac{2|q|^{1/2}+2|q|+\frac{2q^9}{1-q^{16}}+\frac{2q^{7.5}}{1-q^{19}}}{2-(2|q|^3+2|q|^4+\frac{2q^{16}}{1-q^{20}}+\frac{2q^{14}}{1-q^{19}})}\leq 1$$

for $\mathrm{Im}(\tau)>0.345$, which proves the lemma. □

We are now ready to prove:

Theorem 3.4. *Let (x_n,y_n,z_n,t_n) be the optimal F sequence for $\theta_{00}^2(z,\tau),\theta_{01}^2(z,\tau),\theta_{00}^2(0,\tau),\theta_{01}^2(0,\tau)$. For any τ such that $\mathrm{Im}(\tau)\geq 0.345$ and z satisfying the first condition of (16) we have*

$$(x_n,y_n,z_n,t_n)=\left(\theta_{00}^2(z,2^n\tau),\theta_{01}^2(z,2^n\tau),\theta_{00}^2(0,2^n\tau),\theta_{01}^2(0,2^n\tau)\right)$$

Proof. This is true for $n=0$; we prove the statement inductively. Suppose it is true for $n=k$. We have for any τ:

$$\theta_{00}(0,\tau)=1+2q+c,\qquad |c|\leq\frac{2|q|^4}{1-|q|^5}$$

For any τ such that $\mathrm{Im}(\tau)\geq 0.345$, $2|q|\leq 0.676$ and $|c|\leq 0.027$; hence $\mathrm{Re}(\theta_{00}(0,2^k\tau))>0$ for any k, which proves that $\sqrt{z_k}=\theta_{00}(0,2^k\tau)$. Lemma 3.3 shows that $\mathrm{Re}\left(\frac{\theta_{01}(0,\tau)}{\theta_{00}(0,\tau)}\right)>0$, and we also have $\mathrm{Re}\left(\frac{\sqrt{t_k}}{\sqrt{z_k}}\right)\geq 0$ since the choice of roots is good, hence $\sqrt{t_k}=\theta_{01}(0,2^k\tau)$. Proposition 2.13 then proves that $t_{k+1}=\theta_{01}^2(0,2^{k+1}\tau)$ and $z_{k+1}=\theta_{00}^2(0,2^{k+1}\tau)$.

Similarly, given that z satisfies the first condition of (16):

$$|\theta_{00}(z,\tau)-1|\leq|q|^{1/2}+|q|+|q|^3+|q|^4+|q|^{7/2}+|q|^9+\frac{2|q|^{14}}{1-|q|^2}\tag{18}$$

For $\mathrm{Im}(\tau)\geq 0.345$, this is strictly smaller than 1; hence $\mathrm{Re}(\theta_{00}(z,\tau))>0$, which proves that $\sqrt{x_k}=\theta_{00}(z,2^k\tau)$. Again, Lemma 3.3 proves that $\mathrm{Re}\left(\frac{\theta_{01}(z,2^k\tau)}{\theta_{00}(z,2^k\tau)}\right)>0$, and since the choice of signs is good, $\mathrm{Re}\left(\frac{\sqrt{y_k}}{\sqrt{x_k}}\right)\geq 0$, necessarily $\sqrt{y_k}=\theta_{01}(z,2^k\tau)$. This along with Proposition 3.1 finishes the induction. □

Note that a consequence of this proposition is the following fact:

Proposition 3.5. *The optimal F sequence for $\theta_{00}^2(z,\tau),\theta_{01}^2(z,\tau),\theta_{00}^2(0,\tau),\theta_{01}^2(0,\tau)$ converges quadratically to $(1,1,1,1)$.*

3.3 A function with quasi-optimal time evaluation

The strategy of [6] is to use an homogenization of the AGM to get a function $f_\tau:\mathbb{C}\to\mathbb{C}$, on which Newton's method can be applied. To generalize this, we homogenize the function which maps to (x,y,z,t) the limit of the optimal F sequence associated to them; it becomes a function from $\mathbb{C}^2$ to $\mathbb{C}^2$. We call this function F^∞; this function is a major building block for the function we use to compute our two parameters z,τ using Newton's method.

Proposition 3.6. *Let $\lambda,\mu\in\mathbb{C}$. Let $((x_n,y_n,z_n,t_n))_{n\in\mathbb{N}}$ be the optimal F sequence for (x,y,z,t), and $((x'_n,y'_n,z'_n,t'_n))_{n\in\mathbb{N}}$ the optimal F sequence for $(\lambda x,\lambda y,\mu z,\mu t)$. Put $\lim_{n\to\infty}z_n=z_\infty$ and $\lim_{n\to\infty}z'_n=z'_\infty$. Then we have*

$$\mu=\frac{z'_\infty}{z_\infty},\qquad\qquad \lambda=\frac{\left(\lim_{n\to\infty}\left(\frac{x'_n}{z'_\infty}\right)^{2^n}\right)\times z'_\infty}{\left(\lim_{n\to\infty}\left(\frac{x_n}{z_\infty}\right)^{2^n}\right)\times z_\infty}$$

Proof. We prove by induction that

$$x'_n = \epsilon_n \lambda^{1/2^n} \mu^{1-1/2^n} x_n, \quad y'_n = \epsilon_n \lambda^{1/2^n} \mu^{1-1/2^n} y_n, \qquad z'_n = \mu z_n, \quad t'_n = \mu t_n,$$

where $\mathrm{Re}(\lambda^{1/2^n}) \geq 0$, $\mathrm{Re}(\mu^{1-1/2^n}) \geq 0$, and ϵ_n is a 2^n-th root of unity. This is enough to prove the proposition above, since then

$$\lim_{n\to\infty} \left(\frac{x'_n}{z'_\infty}\right)^{2^n} = \lim_{n\to\infty} \lambda \mu^{2^n-1} \left(\frac{x_n}{z'_\infty}\right)^{2^n} = \frac{\lambda}{\mu} \lim_{n\to\infty} \left(\frac{x_n}{z_\infty}\right)^{2^n}.$$

Since this is true for $n = 0$, suppose this is true for $n = k$. We have

$$z'_{k+1} = \frac{z'_k + t'_k}{2} = \mu z_{k+1}.$$

As for t_{k+1}, we can write

$$\sqrt{z'_k} = \epsilon_z \sqrt{\mu} \sqrt{z_k}, \qquad \sqrt{t'_k} = \epsilon_t \sqrt{\mu} \sqrt{t_k}$$

where $\epsilon_z = \pm 1$ and $\epsilon_t = \pm 1$, and the square roots are taken with positive real part. But since $\mathrm{Re}\left(\frac{\sqrt{t'_k}}{\sqrt{z'_k}}\right) \geq 0$ and $\mathrm{Re}\left(\frac{\sqrt{t_k}}{\sqrt{z_k}}\right) \geq 0$, we have $\epsilon_z = \epsilon_t$. Hence

$$t'_{k+1} = \left(\epsilon_z \sqrt{\mu} \sqrt{z_k}\right)\left(\epsilon_z \sqrt{\mu} \sqrt{t_k}\right) = \mu t_{k+1}.$$

As for the other coordinates, we have

$$\sqrt{x'_k} = \epsilon_x \sqrt{\epsilon_k} \lambda^{1/2^{k+1}} \mu^{1/2-1/2^{k+1}} \sqrt{x_k}, \qquad \sqrt{y'_k} = \epsilon_y \sqrt{\epsilon_k} \lambda^{1/2^{k+1}} \mu^{1/2-1/2^{k+1}} \sqrt{y_k}$$

where the roots are taken with positive real part, and $\epsilon_x, \epsilon_y \in \{-1, 1\}$. Since $\mathrm{Re}\left(\frac{\sqrt{y_k}}{\sqrt{x_k}}\right) \geq 0$ and we require $\mathrm{Re}\left(\frac{\sqrt{y'_k}}{\sqrt{x'_k}}\right) \geq 0$, necessarily $\epsilon_x = \epsilon_y$; hence

$$x'_{k+1} = \frac{\sqrt{x'_k}\sqrt{z'_k} + \sqrt{y'_k}\sqrt{t'_k}}{2} = \epsilon_{k+1} \lambda^{1/2^{k+1}} \mu^{1-1/2^{k+1}} \frac{\sqrt{x_k}\sqrt{z_k} + \sqrt{y_k}\sqrt{t_k}}{2} = \epsilon_{k+1} \lambda^{1/2^{k+1}} \mu^{1-1/2^{k+1}} x_{k+1}$$

where $\epsilon_{k+1} = \epsilon_x \sqrt{\epsilon_z}$ is indeed such that $\epsilon_{k+1}^{2^{k+1}} = 1$. This proves the proposition. □

In the case of theta-functions, however, we have:

Proposition 3.7.

$$\lim_{n\to\infty} \frac{\theta(z, 2^n\tau)^{2^n}}{\theta(0, 2^n\tau)^{2^n}} = 1$$

Proof. It is enough to prove $\lim_{n\to\infty} \theta(z, 2^n\tau)^{2^n} = 1$, since it also covers the case $z = 0$. Write, like Equation (18):

$$\theta(z, 2^n\tau) = 1 + q^{2^n}(w^2 + w^{-2}) + c, \qquad |c| \leq \frac{2|q|^{2^n+3}}{1-|q|}$$

We can write $\theta(z, 2^n\tau) = 1 + d_n$ with $|d_n| \leq 2|q|^{2^n}|w^2 + w^{-2}|$. We then have classically

$$|\theta(z, 2^n\tau)^{2^n} - 1| \sim |2^n d_n|$$

and since $\lim_{n\to\infty} 2^n |q|^{2^n} = 0$, this proves the proposition. □

Combining this with Theorems 3.6 and 3.4 proves that, for (x_n, y_n, z_n, t_n) as in Theorem 3.4, we have

$$\lambda = \lim_{n\to\infty} \left(\frac{x'_n}{\mu}\right)^{2^n} \times \mu.$$

In particular, if we define the following function:

$$\begin{aligned} F^\infty : \mathbb{C}^4 &\to \mathbb{C}^2 \\ (x, y, z, t) &\mapsto \left(\left(\lim_{n\to\infty} \left(\frac{x_n}{z_\infty}\right)^{2^n}\right) \times z_\infty, z_\infty\right) \end{aligned}$$

where $z_\infty = \lim_{n\to\infty} z_n$, then we have that, for any z, τ satisfying the hypotheses of Theorem 3.4,

$$F^\infty\left(\lambda\theta_{00}^2(z,\tau), \lambda\theta_{01}^2(z,\tau), \mu\theta_{00}^2(0,\tau), \mu\theta_{01}^2(0,\tau)\right) = (\lambda, \mu).$$

For instance,

$$F^\infty\left(1, \frac{\theta_{01}^2(z,\tau)}{\theta_{00}^2(z,\tau)}, 1, \frac{\theta_{01}^2(0,\tau)}{\theta_{00}^2(0,\tau)}\right) = \left(\frac{1}{\theta_{00}^2(z,\tau)}, \frac{1}{\theta_{00}^2(0,\tau)}\right). \tag{19}$$

This is similar to Equation (11), and will play a similar role in the computation of $\theta(z, \tau)$.

3.4 Convergence

Let us start by showing that, contrary to the AGM and despite Proposition 3.5, an optimal F sequence does not always converge quadratically; for instance, the optimal F sequence for $(2, 2, 1, 1)$ is $((2^{1/2^n}, 2^{1/2^n}, 1, 1))_{n\in\mathbb{N}}$, which does not converge quadratically. This is a big difference from the AGM, and this is why we are reluctant to call optimal F sequences a "generalization of the AGM". However, we now show that the sequence $(\lambda_n) = \left(\left(\frac{x_n}{z_\infty}\right)^{2^n} \times z_\infty\right)_{n\in\mathbb{N}}$ converges quadratically, whence F^∞ can be computed in $O(\mathcal{M}(P)\log P)$ bit operations.

In all that follows, we take $C > 1$ such that $|x_0|, |y_0|, |z_0|, |t_0| \le C$. Note that the equations defining F can then be used to prove inductively that

$$\forall n \in \mathbb{N}, |x_n|, |y_n|, |z_n|, |t_n| \le C.$$

We prove in Section 4.3.4 that there is a suitable C for any z, τ we consider in our final algorithm. We also have a lower bound:

Proposition 3.8. *There exists ϵ, n_0 such that for all $n \ge n_0$, $|x_n|, |y_n|, |z_n|, |t_n| \ge \epsilon$.*

Proof. Note that [4, Prop 2.1] proves that $|z_n|, |t_n| \ge c$, with $c = \frac{2}{\pi}\min(|z_0|, |t_0|)$. Suppose $|x_n|, |y_n|, |z_n|, |t_n| \ge \delta > 0$, and write

$$|\sqrt{x_n} + \sqrt{y_n}|^2 = 2|\sqrt{x_n}|^2 + 2|\sqrt{y_n}|^2 - |\sqrt{x_n} - \sqrt{y_n}|^2.$$

Using $|\sqrt{x_n} - \sqrt{y_n}| \le |\sqrt{x_n} + \sqrt{y_n}|$, which comes the fact that choices of square roots are good, we derive the two following lower bounds.

$$\begin{aligned} |\sqrt{x_n} + \sqrt{y_n}| &\ge \sqrt{2|\sqrt{x_n}|^2 + 2|\sqrt{y_n}|^2 - |x_n - y_n|}, \\ &\ge 2\sqrt{\delta}\sqrt{1 - \frac{|x_n - y_n|}{4\delta}}, \\ \text{and } |\sqrt{x_n} + \sqrt{y_n}|^2 &\ge 2|\sqrt{x_n}|^2 + 2|\sqrt{y_n}|^2 - |\sqrt{x_n} + \sqrt{y_n}|^2, \\ |\sqrt{x_n} + \sqrt{y_n}|^2 &\ge |\sqrt{x_n}|^2 + |\sqrt{y_n}|^2 \ge 2\delta. \end{aligned}$$

Using the same arguments, we obtain:

$$|\sqrt{z_n}+\sqrt{t_n}| \geq 2\sqrt{c}\sqrt{1-\frac{|z_n-t_n|}{4c}},$$
$$|\sqrt{z_n}+\sqrt{t_n}| \geq \sqrt{2c}.$$

We then have

$$|\sqrt{x_n}-\sqrt{y_n}| \leq \frac{|x_n-y_n|}{|\sqrt{x_n}+\sqrt{y_n}|} \leq \frac{|x_n-y_n|}{\sqrt{2\delta}},$$
$$|\sqrt{z_n}-\sqrt{t_n}| \leq \frac{|z_n-t_n|}{|\sqrt{z_n}+\sqrt{t_n}|} \leq \frac{|z_n-t_n|}{\sqrt{2c}}$$

which we use along with Equation 15 to write:

$$|x_{n+1}| \geq \sqrt{\delta}\sqrt{c}\sqrt{\left(1-\frac{|x_n-y_n|}{4\delta}\right)\left(1-\frac{|z_n-t_n|}{4c}\right)} - \frac{|x_n-y_n||z_n-t_n|}{8\sqrt{\delta}\sqrt{c}}$$

Now, using only [4, Prop 2.1] and Equation 15, we can write

$$|x_{n+1}-y_{n+1}| = \frac{|\sqrt{x_n}-\sqrt{y_n}||\sqrt{z_n}-\sqrt{t_n}|}{2} \leq \sqrt{\frac{C}{2c}}|z_n-t_n|$$

which proves that $(|x_n-y_n|)$ converges quadratically to 0; we thus have $|x_n-y_n| \leq C_n\frac{\delta}{2^{2^n}}$, $|z_n-t_n| \leq C'_n\frac{c}{2^{2^n}}$ with $C_n, C'_n = o(2^{2^n})$. This gives

$$|x_{n+1}| \geq \sqrt{\delta}\sqrt{c}\left(\sqrt{\left(1-\frac{C_n}{2^{2^n+2}}\right)\left(1-\frac{C'_n}{2^{2^n+2}}\right)} - \frac{C_n}{2^{2^n+3}}\frac{C'_n}{2^{2^n+3}}\right)$$

Finally, supposing that $c > \delta$, we get that $|x_n| \geq \delta \Rightarrow |x_{n+1}| \geq \delta\epsilon_n$; since (ϵ_n) converges to 1 quadratically, $\prod_{k\geq n}\epsilon_n$ converges to $\epsilon > 0$, which proves that $|x_{n+k}| \geq \epsilon\delta$ for all k. □

We now prove that $(\lambda_n) = \left(\left(\frac{x_n}{z_\infty}\right)^{2^n} \times z_\infty\right)_{n\in\mathbb{N}}$ converges quadratically, by proving the following theorem:

Theorem 3.9. *The sequence (λ_n) converges, to a limit λ. Furthermore, for P large enough, there exists a constant $c_1 > 0$, depending on C, ϵ and $|\lambda|$, such that, if k is the first integer such that $|z_k - t_k| \leq 2^{-P-k-c_1}$, then λ_{k+1} is an approximation of λ with absolute precision P bits.*

Proof. The point here is that once z_n and t_n are close enough, x_{n+1} and y_{n+1} are also close and the value of λ_n does not change much after that. Let $c_1 \geq 0$, and take n the first integer for which $|z_n - t_n| \leq \eta$ with $\eta = 2^{-P-c_1-n}$. We then have for all $k \geq 0$ [6, Theorem 1]:

$$|z_{n+k} - t_{n+k}| \leq A^{2^k-1}\eta^{2^k}$$

with $A = \frac{\pi}{8\min(|z_0|,|t_0|)}$. Furthermore, $|z_{n+1} - z_n| = \frac{1}{2}|z_n - t_n|$, so that

$$|z_\infty - z_{n+k}| \leq \frac{1}{2}\sum_{i=k}^{\infty} A^{2^i-1}\eta^{2^i}$$

and we have $|z_\infty - z_{n+k}| \leq \frac{1}{A}(A\eta)^{2^k}$. Finally, using Equation (15), one can write

$$\begin{aligned}
|x_{n+k+1} - y_{n+k+1}| &\leq \frac{|\sqrt{x_n} - \sqrt{y_n}||\sqrt{z_n} - \sqrt{t_n}|}{2} \\
&\leq \sqrt{C}\sqrt{2}\sqrt{|z_{n+k+1} - t_{n+1}|} \qquad \text{since } z_{n+k+1} - t_{n+k+1} = \frac{(\sqrt{z_{n+k}} - \sqrt{t_{n+k}})^2}{2} \\
&\leq \sqrt{2AC}|z_{n+k} - t_{n+k}|.
\end{aligned}$$

Now, define $q_n = \frac{(x_n/z_\infty)^2}{x_{n-1}/z_\infty}$, so that $\frac{\lambda_{n+1}}{\lambda_n} = q_n^{2^n}$. Note that if one makes the approximation $x_{n+k+1} = y_{n+k+1}$ and $z_{n+k+1} = t_{n+k+1} = z_\infty$, we have $x_{n+k+2} = \sqrt{x_{n+k+1}z_\infty}$ which gives $q_{n+k+2} = 1$. We take a closer look at those approximations:

$$\begin{aligned}
|x_{n+k+2} - \sqrt{x_{n+k+1}}\sqrt{z_{n+k+1}}| &\leq \frac{|\sqrt{y_{n+k+1}} - \sqrt{x_{n+k+1}}||\sqrt{z_{n+k+1}} + \sqrt{t_{n+k+1}}|}{4} \\
&\quad + \frac{|\sqrt{y_{n+k+1}} + \sqrt{x_{n+k+1}}||\sqrt{z_{n+k+1}} - \sqrt{t_{n+k+1}}|}{4} \\
&\leq \frac{\sqrt{C}}{2}(|\sqrt{y_{n+k+1}} - \sqrt{x_{n+k+1}}| + |\sqrt{z_{n+k+1}} - \sqrt{t_{n+k+1}}|) \\
&\leq \frac{\sqrt{C}}{2}\left(\frac{\sqrt{2AC}}{|\sqrt{x_{n+k}} + \sqrt{y_{n+k}}|}|z_{n+k} - t_{n+k}| + \sqrt{2A}|z_{n+k+1} - t_{n+k+1}|\right) \\
&\leq C'(A\eta)^{2^k}
\end{aligned}$$

so

$$\begin{aligned}
q_{n+k+2} - 1 &= \frac{(x_{n+k+2}/z_\infty)^2 - x_{n+k+1}/z_\infty}{x_{n+k+1}/z_\infty} \\
&\leq \frac{\frac{1}{z_\infty^2}(\sqrt{x_{n+k+1}}\sqrt{z_{n+k+1}} + C'(A\eta)^{2^k})^2 - x_{n+k+1}/z_\infty}{x_{n+k+1}/z_\infty} \\
&\leq \frac{x_{n+k+1}z_{n+k+1}/z_\infty^2 - x_{n+k+1}/z_\infty}{x_{n+k+1}/z_\infty} + \frac{2C/z_\infty^2 C'(A\eta)^{2^k} + C'^2(A\eta)^{2^{k+1}}}{x_{n+k+1}/z_\infty} \\
&\leq \frac{1}{2z_\infty}\sum_{i=k+1}^{\infty}(A\eta)^{2^i} + \frac{CC'(A\eta)^{2^k}}{x_{n+k+1}z_\infty} + \frac{C'^2(A\eta)^{2^{k+1}}}{x_{n+k+1}/z_\infty} \\
&\leq c \times (A\eta)^{2^k}
\end{aligned}$$

This proves that (q_n) converges quadratically to 1; using the equivalent $q_n^{2^n} - 1 \sim 2^n q_n$, we have that $(q_n^{2^n})$ also converges quadratically to 1, which proves the convergence of the sequence (λ_n).

Finally we have

$$\begin{aligned}
q_{n+2}^{2^{n+2}} \ldots q_{n+k}^{2^{n+k}} - 1 &\leq \prod_{i=0}^{k-2}\left(1 + c(A\eta)^{2^i}\right)^{2^{n+i+2}} - 1 \\
&< \exp\left(4c\sum_{i=0}^{k-2}2^{n+i}(A\eta)^{2^i}\right) - 1 \\
&\leq \frac{4c\sum_{i=0}^{k-2}2^{n+i}(A\eta)^{2^i}}{1 - (4c\sum_{i=0}^{k-2}2^{n+i}(A\eta)^{2^i})/2}
\end{aligned}$$

which proves, for P large enough,

$$\begin{aligned} |\lambda - \lambda_{n+1}| &\leq 8c|\lambda_{n+1}| \sum_{i=0}^{\infty} 2^{n+i}(A\eta)^{2^i} \\ &\leq 16c|\lambda_{n+1}|A\eta 2^n \\ &\leq 16Ac|\lambda_{n+1}|2^{-c_1} \times 2^{-P} \end{aligned}$$

This inequality proves that, at least for P large enough, $|\lambda_{n+1}| \leq 2|\lambda|$. Hence if we suppose $\log_2(32Ac|\lambda|) \leq c_1$, we have that λ_{n+1} is an approximation of λ with P bits of absolute precision. □

Algorithm 2 Compute $F^\infty(x, y, z, y)$

1: Work at precision $\mathcal{P}$.
2: $n \leftarrow 0$
3: **while** $|z - t| \leq 2^{-P-n-c_1}$ **do**
4: $\quad n \leftarrow n + 1$
5: $\quad (x, y, z, t) \leftarrow F(x, y, z, t)$
6: **end while**
7: $(x, y, z, t) \leftarrow F(x, y, z, t)$
8: Return $\left(\left(\frac{x}{z}\right)^{2^{n+1}} \times z, z\right)$

This gives an algorithm, Algorithm 2, to compute $F^\infty(x, y, z, t)$. According to [6, Theorem 12], if $n = \max(\log|\log|z_0/t_0||, 1) + \log(P + c_1)$, a_n is an approximation of AGM $\left(1, |\frac{z_0}{t_0}|\right)$ with relative precision P bits. This proves that at the end of the algorithm, $n = O(\log P)$; in fact, we have more precisely $n \leq \log_2 P + C''$ with C'' a constant independent of P. Finally in the next subsection proves that one can take $\mathcal{P} = P + O(\log P)$, which means that this algorithm computes F^∞ in $O(\mathcal{M}(P) \log P)$ bit operations.

3.5 Loss of precision

We use Theorem 2.9 in order to evaluate the precision lost when computing $F^\infty(x, y, z, t)$. First note that the upper and lower bounds on the terms of the sequence allow us to write

$$\begin{aligned} \left(\frac{1}{\sqrt{|z_n|}} + \frac{1}{\sqrt{|t_n|}}\right)\left(\sqrt{|z_n|} + \sqrt{|t_n|}\right) &\leq b/2 \\ \left(\frac{1}{\sqrt{|z_n|}} + \frac{1}{\sqrt{|t_n|}}\right)\left(\sqrt{|x_n|} + \sqrt{|y_n|}\right) &\leq b \\ \left(\frac{1}{\sqrt{|x_n|}} + \frac{1}{\sqrt{|y_n|}}\right)\left(\sqrt{|z_n|} + \sqrt{|t_n|}\right) &\leq b \end{aligned}$$

for some $b > 1$; for instance, one can take $b = \max\left(1, 4\sqrt{\frac{C}{\epsilon}}\right)$. We prove in Section 4.3.4 the existence of ϵ and C, and hence of b, for any values of theta we consider as arguments.

We first evaluate a bound on the error incurred when computing F using Equation (15). Using those formulas allows us to get error bounds that are identical for F_x and F_y, and F_z and

F_t. For simplicity, we assume that the error on z and t is the same, as well as the error on x and y. This gives:

$$\begin{aligned}
|\operatorname{Re}(F_x) - \operatorname{Re}(\tilde{F}_x)| &\leq \left(1 + k_z\left(\frac{1}{\sqrt{|z|}} + \frac{1}{\sqrt{|t|}}\right)\left(\sqrt{|x|} + \sqrt{|y|}\right) + k_x\left(\frac{1}{\sqrt{|x|}} + \frac{1}{\sqrt{|y|}}\right)\left(\sqrt{|z|} + \sqrt{|t|}\right)\right) 2^{-P} \\
|\operatorname{Re}(F_z) - \operatorname{Re}(\tilde{F}_z)| &\leq \left(1 + 2k_z\left(\frac{1}{\sqrt{|z|}} + \frac{1}{\sqrt{|t|}}\right)\left(\sqrt{|z|} + \sqrt{|t|}\right)\right) 2^{-P}
\end{aligned}$$

We thus get the following induction relations when looking at what happens when applying F n times in a row:

$$k_x^{(n)} \leq 1 + bk_z^{(n-1)} + bk_x^{(n-1)}, \qquad k_z^{(n)} \leq 1 + bk_z^{(n-1)}$$

The last equation rewrites as $k_z^{(n)} + \frac{1}{b-1} \leq b\left(k_z^{(n-1)} + \frac{1}{b-1}\right)$, which gives $k_z^{(n)} \leq b^n\left(k_z + \frac{1}{b-1}\right)$. The induction for x becomes $k_x^{(n)} \leq 1 + (k_z + \frac{1}{b-1})b^n + bk_x^{(n-1)}$, which we solve:

$$k_x^{(n)} \leq (1 + b + ... + b^n)k_x + nb^n\left(k_z + \frac{1}{b-1}\right) \leq b^n\left(nk_z + \frac{b+n}{b-1}\right)$$

For $b > 1$, we have for n large enough that $k^{(n)} \leq 2b^{2n}$, which ultimately means the number of bits lost when applying F n times in a row is bounded by $2n\log b + 1$.

Finally we need to find the number of bits lost in the computation of $\left(\frac{x_n}{z_\infty}\right)^{2^n}$. Call E_k the error made after computing k squarings in a row; we have the following recurrence relation:

$$E_{k+1} \leq 2 + 4E_k|x_n/z_\infty|^{2^k}$$

However, since (λ_n) converges, $|\lambda_n| \leq \rho$ for some constant ρ; furthermore, for any $k \leq n$, one has $|x_n/z_\infty|^{2^k} \leq 1 + \frac{\rho}{z_\infty}$. Hence the recurrence becomes $E_{k+1} \leq 2 + 4\left(1 + \frac{\rho}{z_\infty}\right)E_k$, which we solve to get

$$E_n \leq 2\frac{C'^{n+1} - 1}{C' - 1} \leq \frac{2}{C' - 1}C'^{n+1}$$

with $C' = 4\left(1 + \frac{\rho}{z_\infty}\right)$. This means the number of bits lost after n successive squarings is at the most $(n+1)\log C' + 1 - \log(C' - 1)$.

Overall, if we write that the final value of n in Algorithm 2 is bounded by $\log_2 P + C''$, we have that the number of bits lost is bounded by

$$(2\log_2 b + \log C')(\log_2 P + C'') + \log C' + 2 - \log C' - 1$$

which is $O(\log P)$.

4 Fast computation of θ

We use a similar method as [6], that is to say finding a function $\mathfrak{F}$ such that

$$\mathfrak{F}\left(\frac{\theta_{01}^2(z,\tau)}{\theta_{00}^2(z,\tau)}, \frac{\theta_{01}^2(0,\tau)}{\theta_{00}^2(0,\tau)}\right) = (z,\tau),$$

which can then be inverted using Newton's method. One can then compute $\theta(z,\tau)$ by, for instance, using Equation (19) and extracting a square root, determining the correct choice of sign by computing a low-precision (say, 10 bits) approximation of the value using the naive method; we use a different trick in our final algorithm (Algorithm 5). We build this function $\mathfrak{F}$ using F^∞ as a building block.

4.1 Building $\mathfrak{F}$

Just as with the algorithm for theta-constants, we use formulas derived from the action of $SL_2(\mathbb{Z})$ on the values of θ in order to get multiplicative factors depending on our parameters; this will allow us to build a function which computes z, τ from the values $\theta_i(z,\tau)$. We define the function $\mathfrak{F}$ as the result of Algorithm 3.

Algorithm 3 Compute $\mathfrak{F}(s,t)$

1: $b \leftarrow \sqrt{1-t'^2}$ ▷ Choose the root with positive real part [4, Prop 2.9]
2: $a \leftarrow \frac{1-st}{b}$
3: $(x,y) \leftarrow F^\infty(1,a,1,b)$
4: $(q_1,q_2) \leftarrow F^\infty(1,s,1,t)$
5: Return $\left(\sqrt{\log\left(\frac{q_2 x}{q_1 y}\right) \times \frac{q_2/y}{-2\pi}}, i\frac{q_2}{y}\right)$, choosing the sign of the square root so that it has positive imaginary part.

Proposition 4.1. *Let τ be such that $|\operatorname{Re}(\tau)| \leq 0.5$, $\operatorname{Im}(\tau) \geq 0.345$ and $\operatorname{Im}\left(\frac{-1}{\tau}\right) \geq 0.345$, and let z be such that the conditions (16) are satisfied. Then*

$$\mathfrak{F}\left(\frac{\theta_{01}^2(z,\tau)}{\theta_{00}^2(z,\tau)}, \frac{\theta_{01}^2(0,\tau)}{\theta_{00}^2(0,\tau)}\right) = (z,\tau)$$

Proof. Equation (19) proves that $(q_1,q_2) = \left(\frac{1}{\theta_{00}(z,\tau)^2}, \frac{1}{\theta_{00}(0,\tau)^2}\right)$. Furthermore, using Jacobi's formula (5) and the equation defining the variety (6), it is easy to see that $b = \frac{\theta_{10}(0,\tau)^2}{\theta_{00}(0,\tau)^2}$ and $a = \frac{\theta_{10}(z,\tau)^2}{\theta_{00}(z,\tau)^2}$.

The formulas in [12, Table V, p.36] give

$$\left(\theta_{00}^2(z,\tau), \theta_{10}^2(z,\tau), \theta_{00}^2(0,\tau), \theta_{10}^2(0,\tau)\right) = \left(\lambda\theta_{00}^2\left(\frac{z}{\tau}, \frac{-1}{\tau}\right), \lambda\theta_{01}^2\left(\frac{z}{\tau}, \frac{-1}{\tau}\right), \mu\theta_{00}^2\left(0, \frac{-1}{\tau}\right), \mu\theta_{01}^2\left(0, \frac{-1}{\tau}\right)\right)$$

with $\lambda = \frac{e^{-2i\pi z^2/\tau}}{-i\tau}$, $\mu = \frac{1}{-i\tau}$. From the discussion in Section 3.2.1, the conditions on z, τ allow us to apply Theorem 3.4 to $\frac{z}{\tau}, \frac{-1}{\tau}$. This proves that

$$F^\infty\left(\theta_{00}^2(z,\tau), \theta_{10}^2(z,\tau), \theta_{00}^2(0,\tau), \theta_{10}^2(0,\tau)\right) = \left(\frac{e^{-2i\pi z^2/\tau}}{-i\tau}, \frac{1}{-i\tau}\right),$$

and by homogeneity, $(x,y) = \left(\frac{e^{-2i\pi z^2/\tau}}{-i\tau\theta_{00}(z,\tau)^2}, \frac{1}{-i\tau\theta_{00}(0,\tau)^2}\right)$. □

This means that, starting from the knowledge of z and τ with precision P and a low-precision approximation of the quotients $\frac{\theta_{01}(z,\tau)}{\theta_{00}(z,\tau)}$ and $\frac{\theta_{01}(0,\tau)}{\theta_{00}(0,\tau)}$, one can compute those quotients with precision P using Newton's method. This is Algorithm 4.

We make a few remarks:

- Much in the same way as [9], we find it preferable to use finite differences to compute the coefficients a_{11}, a_{21}, a_{22} of the Jacobian, as it does not require the computation of the derivative of $\mathfrak{F}$, which could be tedious.

Algorithm 4 Compute $\theta_{00}^2(z,\tau), \theta_{01}^2(z,\tau), \theta_{00}^2(0,\tau), \theta_{01}^2(0,\tau)$ with precision P.
Input: (z,τ) with absolute precision P.

1: Compute $\theta_{00,01}^2(z,\tau)$, $\theta_{00,01}^2(0,\tau)$ with absolute precision P_0 using Algorithm 1.
2: $s \leftarrow \frac{\theta_{01}(z,\tau)^2}{\theta_{00}(z,\tau)^2}$, $t \leftarrow \frac{\theta_{01}(0,\tau)^2}{\theta_{00}(0,\tau)^2}$
3: $p \leftarrow P_0$
4: **while** $p \leq \mathcal{P}'$ **do**
5: $\quad p \leftarrow 2p$
6: $\quad$ Compute $a_{11} = \frac{\partial \mathfrak{F}_x}{\partial x}(s,t)$, $a_{22} = \frac{\partial \mathfrak{F}_y}{\partial y}(s,t)$, $a_{12} = \frac{\partial \mathfrak{F}_x}{\partial y}(s,t)$ with precision p.
7: $\quad (s,t) \leftarrow (s,t) - (\mathfrak{F}(s,t) - (z,\tau)) \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}^{-1}$
8: **end while**
9: $(a,b) \leftarrow F^\infty(1,s,1,t)$
10: $(a,b) \leftarrow (1/a, 1/b), \qquad (s,t) = (sa, tb)$
11: Return (a,s,b,t).

- The value of P_0 has to be large enough that Newton's method converges. We note that, in general, a lower bound on P_0 may depend on the arguments; for instance, [6] experimentally finds $4.53\,\mathrm{Im}(\tau)$ to be a suitable lower bound for P_0 when computing theta-constants. However, we outline in the next section a better algorithm which only uses the present algorithm for z,τ within a compact set; hence, P_0 can be chosen to be a constant, and we use in practice $P_0 = 30000$.

We do not provide a full analysis for this algorithm: we outline in the next section a better algorithm, which uses this algorithm as a subroutine, and we will provide a full analysis at that time. It is enough to say that the computation of F^∞ at precision p is done in time $O(\mathcal{M}(p)\log p)$ using Algorithm 2; however, this running time depends on z,τ, since it depends on the bounds C,ϵ that one can write for $|x_n|, |y_n|, |z_n|, |t_n|$. Hence, the cost of evaluating $\mathfrak{F}$ at precision p is $O(\mathcal{M}(p)\log p)$ bit operations, and the fact that we double the working precision at every step means that the algorithm is as costly as the last iteration. Furthermore, one should choose $\mathcal{P}'$ so that the final result is accurate with absolute precision P. This means compensating the loss of absolute precision incurred during the computation of $\mathfrak{F}$; in general, this only depends on $\mathrm{Im}(\tau)$ and linearly in $\log p$. Furthermore, we have the following proposition:

Proposition 4.2. *Let x and y be approximations of a and b with absolute precision N, and x', y' the result of Step 7 in Algorithm 4 when using finite differences to approximate the Jacobian matrix. Then x', y' are approximations of a, b with precision $2P_0 - \delta$.*

Its proof can be adapted from the proof of [9, Theorem 12]. In practice, we found $\delta = 4$ to be enough. Determining the number of bits lost at each step can be done in the same way as [9, p. 19]: if $s^{(n-1)}$ and $s^{(n-2)}$ agree to k bits, and $s^{(n)}$ and $s^{(n-1)}$ agree to k' bits, the number of bits lost can be computed as $2k - k'$. In the end, working at precision $\mathcal{P}' = P + c\log P + d$, with c, d independent of P but functions of z,τ, is enough to compensate all precision losses; this proves that the running time of this algorithm is asymptotically $O(\mathcal{M}(P)\log P)$.

4.2 Computing $\theta(z,\tau)$ in uniform quasi-optimal time

We now show an algorithm with uniform (i.e. independent in z and τ) quasi-optimal complexity that computes $\theta(z,\tau)$ for any (z,τ) satisfying conditions (3). We use the same strategy as [6];

namely, we use the naive algorithm when $\mathrm{Im}(\tau)$ is large; and for smaller values of $\mathrm{Im}(\tau)$, we put $\tau' = \frac{\tau}{2^s}$ so that τ' is within a compact set, then use Algorithm 4, which complexity will be uniform since its arguments belong to a compact set. However we also need to divide z by a power of 2 so that it also belongs to a compact set, and so that (z', τ') satisfies conditions (3) and (16). Once $\theta\left(\frac{z}{2^t}, \frac{\tau}{2^s}\right)$ has been computed by the previous algorithm, we alternate between using Equation (12) to double the second argument and Equation (17) to double the first argument, until finally recovering $\theta(z, \tau)$. This is Algorithm 5.

Algorithm 5 Compute $\theta(z, \tau)$ for $\tau \in \mathcal{F}$ and z reduced

1: **if** $P \leq 25\,\mathrm{Im}(\tau)$ **then**
2: Compute $\theta_{00,01,10}(z, \tau), \theta_{00,01,10}(0, \tau)$ with precision P using the naive method (Algorithm 1 + Section 2.2.4).
3: **else**
4: Take $s \in \mathbb{N}$ such that $1 \leq |\tau|/2^s < 2$
5: Put $\tau_1 = \frac{\tau}{2^s}$ and $z_1 = \frac{z}{2^s}$, so that $\mathrm{Im}(z_1) \leq \mathrm{Im}(\tau_1)/2$.
6: Put $z_2 = z_1/4$ and $\tau_2 = \tau_1/2$.
7: Compute approximations of absolute precision $\mathcal{P}$ of $\theta_{00}^2(z_2, \tau_2), \theta_{01}^2(z_2, \tau_2)$, $\theta_{00}^2(0, \tau_2)$, and $\theta_{01}^2(0, \tau_2)$ using Algorithm 4.
8: Compute $\theta_{00}^2(z_2, \tau_1), \theta_{01}^2(z_2, \tau_1), \theta_{00}^2(0, \tau_1), \theta_{01}^2(0, \tau_1)$ using Equation (12), and $\theta_{10}^2(z_2, \tau_1)$ using Equation (14) and $\theta_{10}^2(0, \tau_1)$ using its equivalent in $z = 0$.
9: Compute $\theta_{00,01,10}(0, \tau_1)$.
10: Compute $\theta_{00,01}(z_1/2, \tau_1)$ using Equation (17).
11: **for** i = 1 .. s **do**
12: Compute $\theta_{00}^2(0, 2^i\tau_1), \theta_{01}^2(0, 2^i\tau_1)$ using the AGM.
13: Compute $\theta_{00}^2(2^{i-2}z_1, 2^i\tau_1), \theta_{01}^2(2^{i-2}z_1, 2^i\tau_1)$ using Equation (12).
14: If $i = s$, compute also $\theta_{10}^2(0, 2^i\tau_1)$ using the equivalent of Equation (14) in $z = 0$, then $\theta_{10}(0, 2^i\tau_1)$ by taking the square root.
15: Compute $\theta_{10}^2(2^{i-2}z_1, 2^i\tau_1)$ using Equation (14).
16: Compute $\theta_{00,01}(0, 2^i\tau_1)$.
17: Compute $\theta_{00}(2^{i-1}z_1, 2^i\tau_1), \theta_{01}(2^{i-1}z_1, 2^i\tau_1)$ using Equation (17).
18: **end for**
19: Compute $\theta_{10}^2(2^{s-1}z_1, 2^s\tau_1)$ using Equation (6).
20: Compute $\theta_{00,01,10}(z, \tau)$ using Equation (17).
21: **end if**

A few notes on this algorithm:

- We note that, at several steps of the algorithm (e.g. Steps 9, 14, 16) we need to compute theta-constants from their square. The correct choice of signs is given by the proof of Theorem 3.4, which shows that $\mathrm{Re}(\theta_{00}(0, \tau)) \geq 0$ and $\mathrm{Re}(\theta_{01}(0, \tau)) \geq 0$; and furthermore, since $\mathrm{Re}(q^{1/4}) \geq |q|^{1/4}\cos(\pi/8)$, we also have $\mathrm{Re}(\theta_{10}(0, \tau)) \geq 0$.
- Taking $\tau_2 = \tau_1/2$ allows us to use Equation (14) in step 8 instead of Equation (5) and Equation (6), which is more efficient and loses fewer bits.
- The knowledge of $\theta_{10}^2(2^{i-2}z_1, 2^i\tau)$ is enough for the z-duplication formulas of step 17, and it can be computed directly from θ_{00} and θ_{01} using Equation (14).
- Computing $\theta_{11}(z, \tau)$ is also possible; one should use a partial summation if $P \leq 25\,\mathrm{Im}(\tau)$. In the other case, since all the z-duplication formulas for $\theta_{11}(z, \tau)$ involve a division by

$\theta_{10}(0,\tau)$ [12, p.22], it is just as efficient to simply use Equation (7) after Step 20, then extract the square root. The square root extraction loses $O(\mathrm{Im}(\tau)) = O(P)$ bits, and this also gives a quasi-optimal algorithm.

4.3 Proving the correctness of the algorithm

This section is devoted to proving the following theorem:

Theorem 4.3. *For any τ, z satisfying conditions (3), Algorithm 5 with $\mathcal{P} = 2P$ computes $\theta_{00,01,10}(z,\tau), \theta_{00,01,10}(0,\tau)$ with absolute precision P in $O(\mathcal{M}(P)\log P)$ bit operations.*

As we discussed in Section 2, this also gives an algorithm that computes $\theta(z,\tau)$ for any $(z,\tau) \in \mathbb{C} \times \mathcal{H}$; one simply needs to reduce τ in $\tau' \in \mathcal{F}$, then reduce z in z', and deduce $\theta(z,\tau)$ from $\theta(z',\tau')$ using Equations (1) and (2). This causes a loss of absolute precision which depends on z and τ, and this algorithm is no longer uniform.

We need to perform an analysis of the number of bits lost by the algorithm; once again, we use Theorem 2.9. For each step, we proceed as follows: assuming the error on all the quantities is bounded by k, determine a factor x such that the error on the quantities we get after the computation is bounded by xk, then declare the number of bits lost in this step to be $\log x$; this gives a very loose upper bound, but simplifies the process.

Finally, we also need to prove that the hypotheses made in Sections 3.4 and 3.5 are verified in Step 7 of the algorithm. This is necessary to prove that the sequence (λ_n) we consider is quadratically convergent, and that the number of bits lost is only $O(\log P)$. We prove this in Section 4.3.4, which then completes the proof that the running time is indeed uniform and quasi-optimal.

4.3.1 Naive algorithm

As we showed in Theorem 2.8, the number of bits lost when using the naive algorithm is $\log B + 7$, although this constant could be made even smaller when taking into account that $P \leq 25\,\mathrm{Im}(\tau)$. Furthermore, $\sqrt{\frac{P}{\mathrm{Im}(\tau)}} \leq 25$, which means the running time of this step is asymptotically dominated by the cost of the computation of π, q and w with precision $\mathcal{P} = P + \log B + 7$, which takes $O(\mathcal{M}(P)\log P)$ bit operations.

4.3.2 Square root extraction

Steps 9, 16 and 14 require extracting square roots, which multiply the error by $\frac{1}{\sqrt{|z|}}$. We prove in the next subsection that $|\theta_{00,01}(0, 2^i\tau_1)| \geq 0.859$ for $i = [1 \dots s]$. Hence, each extraction of square root loses at most 4 bits: step 9 loses 4 bits, and step 16 loses $4s \leq 4\log P$ bits.

Step 14 loses more bits since $\theta_{10}(0,\tau)$ is smaller; indeed, $|\theta_{10}(0,\tau)| \sim |q|^{1/4}$. This means the number of bits lost during this step is bounded by $\frac{\log|q|}{8} = \frac{\pi}{8}\log_2 e\,\mathrm{Im}(\tau)$.

4.3.3 Duplication formulas and finishing the proof of correctness

The algorithm uses both τ-duplication formulas and z-duplication formulas, and we need to analyse how many bits are lost for each application of those formulas.

The τ-duplication formulas are nothing more than applying F to $\theta^2_{00,01}(z,\tau)$ and $\theta^2_{00,01}(0,\tau)$. However, the analysis here is simpler than in section 3.5, because we do not need to compute the square roots of $\theta_{00,01}(z,\tau)$, since they are directly given by step 17. Hence we just need to account for the error of the additions, subtractions and multiplications in Equation (15); since

all the quantities are bounded, this means each step loses a constant number g of bits (our analysis shows that $g \leq 10.48$). In the end, the τ-duplication formulas account for the loss of $g \times s \leq g \log P$ bits of precision.

As for the z-duplication formulas, we need to perform several analyses. Looking at Equation (17), one needs to evaluate the fourth power of theta functions, then add them; then evaluate the third power of theta constants, then perform a division. Computing the error using the formulas from 2.9 is rather straightforward when one has bounds on those quantities, which are given by the following theorem:

Theorem 4.4. *Assume* $\mathrm{Im}(\tau) > \sqrt{3}/2$. *Then*

$$0.859 \leq |\theta_{00,01}(0,\tau)| \leq 1.141, \qquad |\theta_{10}(0,\tau)| \leq 1.018$$

We also have:

- *Suppose that* $0 \leq \mathrm{Im}(z) \leq \frac{\mathrm{Im}(\tau)}{8}$, *as in Steps 10 and 17. Then* $|w|^{-2n} \leq e^{n\pi\,\mathrm{Im}(\tau)/4}$ *and*

$$0.8038 \leq |\theta_{00,01}(z,\tau)| \leq 1.1962 \qquad |\theta_{10}(z,\tau)| \leq 1.228$$

- *Suppose that* $0 \leq \mathrm{Im}(z) \leq \frac{\mathrm{Im}(\tau)}{4}$, *as in Steps 20. Then* $|w|^{-2n} \leq e^{n\pi\,\mathrm{Im}(\tau)/2}$ *and*

$$0.6772 \leq |\theta_{00,01}(z,\tau)| \leq 1.3228 \qquad |\theta_{10}(z,\tau)| \leq 1.543$$

Proof. The bounds on the theta-constants come from [6, p. 5], which proves $|\theta_{00,01}(0,\tau) - 1| \leq \frac{2|q|}{1-|q|}$. The techniques are the same as the proof of Lemma 3.3 or Theorem 2.6. This gives in the first case

$$\begin{aligned}
|\theta_{00,01}(z,\tau) - 1| &\leq |q|^{3/4} + |q| + |q|^{7/2} + |q|^4 + |q|^{8.25} + |q|^9 + \frac{|q|^{15}}{1-|q|} \\
&\leq 0.1962 \qquad \text{since } \mathrm{Im}(\tau) \geq \sqrt{3}/2 \\
|\theta_{10}(z,\tau) - q^{1/4}(w + w^{-1})| &\leq \frac{q^{15/8}}{1-q^{3/8}} \leq 0.009
\end{aligned}$$

so $|\theta_{10}(z,\tau)| \leq |q|^{1/4}(|w| + |w|^{-1}) + 0.009 \leq 1.228$. In the second case:

$$\begin{aligned}
|\theta_{00,01}(z,\tau) - 1| &\leq |q|^{1/2} + |q| + \frac{|q|^3}{1-|q|} \leq 0.3228 \\
|\theta_{10}(z,\tau) - q^{1/4}(w + w^{-1})| &\leq |q|^{5/4} + |q|^{9/4} + ... \leq \frac{q^{5/4}}{1-q} \leq 0.0357
\end{aligned}$$

□

Combining these bounds with formulas from Theorem 2.9 gives the following bounds:

$$\begin{aligned}
\mathrm{error}(\theta_{00}(2^i z_1, 2^{i+1}\tau_1)) &\leq (20050.518 + 1818.032 k_{\theta^z_{01}} + 1966.823 k_{\theta^z_{10}} + 33516 k_{\theta^0_{00}}) 2^{-P} \\
\mathrm{error}(\theta_{01}(2^i z_1, 2^{i+1}\tau_1)) &\leq (20050.518 + 1818.032 k_{\theta^z_{00}} + 1966.823 k_{\theta^z_{10}} + 33516 k_{\theta^0_{01}}) 2^{-P}
\end{aligned}$$

which means losing at most 16 more bits of precision.

Step 19 causes the loss of a greater number of bits. We use Equation (6) instead of the third z-duplication formula, because dividing by $\theta_{10}(0,\tau)^2$ loses less bits than dividing by $\theta_{10}(0,\tau)^3$,

and we only need the knowledge of $\theta_{10}^2(2^{s-1}z, 2^s\tau)$ for the next step anyway. This amounts to computing:

$$\theta_{10}^2(z,\tau) = \frac{\theta_{00}^2(z,\tau)\theta_{00}^2(0,\tau) - \theta_{01}^2(z,\tau)\theta_{01}^2(0,\tau)}{\theta_{10}^2(0,\tau)}$$

Computing the numerator multiplies the error by a factor at most 60, and the norm of this numerator is bounded by 4.557; we then get from Theorem 2.9 that the error is bounded by $\frac{m}{|\theta_{10}(0,\tau)|^8} \sim m|q|^{-2}$, with $m \leq 1600$. In the end, we lose at most $2\pi \log_2 e \operatorname{Im}(\tau) + 11$ bits.

Finally, we also lose a great number of bits during the last application of the z-duplication formulas in step 20, since the formula for $\theta_{10}(z,\tau)$ requires dividing by $\theta_{10}(0,\tau)^3$. The error is thus multiplied by $|q|^{-3}$ up to a constant factor; this means a loss of $3\pi \log_2 e \operatorname{Im}(\tau)$ bits, plus a constant.

In the end, we see that the number of lost bits is bounded by $(2\pi + \pi/8 + 3\pi)\operatorname{Im}(\tau) \log_2 e + c\log P + d$; given that $P \geq 25\operatorname{Im}(\tau)$, and that $5.125\pi \log_2 e \leq 23.3$, the number of bits lost is thus less than P. This means that $\mathcal{P} = 1.93P + c\log P + d \leq 2P$ is enough to get a result which is accurate to absolute precision P; this also means that we indeed never have an error k bigger than $2^{(2P)/2}$, which is necessary to apply Theorem 2.9.

4.3.4 Proof of quadratic convergence and quasi-optimal running time

It remains to prove that the complexity is the right one. If $P \geq 25\operatorname{Im}(\tau)$, $\log_2 P > \log_2 \operatorname{Im}(\tau) + 4.7$, which means $s \leq \log P$ and the cost of Steps 11 to 18 is $O(\mathcal{M}(\mathcal{P})\log P)$. We verify that conditions (3) and (16) hold:

$$|\operatorname{Re}(\tau_2)| \leq 1/2^{s+2} \leq 1/4, \qquad 0.345 \leq \frac{\sqrt{3}}{4} \leq \operatorname{Im}(\tau_2) \leq 1,$$
$$|\operatorname{Re}(z_2)| \leq 1/2^{s+3} \leq 1/8, \qquad 0 \leq \operatorname{Im}(z_2) \leq \frac{\operatorname{Im}(\tau_2)}{4}.$$

This means the choices of signs are always good, and hence our result is indeed (squares of) theta-functions and theta-constants.

We also need prove that there is a $C > 1$ such that, for all z_2, τ_2 that we consider,

$$\frac{\theta_{01}^2(0,\tau_2)}{\theta_{00}^2(0,\tau_2)} \leq C, \quad \frac{\theta_{10}^2(0,\tau_2)}{\theta_{00}^2(0,\tau_2)} \leq C, \qquad \frac{\theta_{01}^2(z_2,\tau_2)}{\theta_{00}^2(z_2,\tau_2)} \leq C, \quad \frac{\theta_{10}^2(z_2,\tau_2)}{\theta_{00}^2(z_2,\tau_2)} \leq C.$$

This is a direct consequence of the fact that z_2, τ_2 are within a compact set that does not contain any zero of $\theta(z,\tau)$; hence one can write (non-zero) lower and upper bounds for any of the values of theta. If one wants to be a bit more precise, using the same reasoning as in Theorem 4.4, we have for $\sqrt{3}/4 \leq \operatorname{Im}(\tau) \leq 1$:

$$|\theta_{00,01}(z,\tau) - 1| \leq |q|^{1/2} + |q| + \frac{|q|^3}{1-|q|} \leq 0.7859, \qquad |\theta_{10}(z,\tau)| \leq 1 + |q|^{1/4} + \frac{|q|^{5/4}}{1-|q|} \leq 1.958.$$

This gives $C \leq 83.64$ and $\epsilon \geq \frac{0.042^2}{1.7859^2} \simeq \frac{1}{1808}$. Furthermore, with a careful analysis, one can prove that $c_1 = 55$ is enough in Theorem 3.9.

In any case, this proves that (λ_n) is quadratically convergent. We note that the fact that z_2, τ_2 are within a compact shows that the constants b_1, b_2, b_3 exist and are independent of z, τ. This makes the running time of Step 7 only dependent in P, which was the point of the uniform algorithm. In particular, the number of bits lost during the computation of F^∞ or in $\mathfrak{F}$ can be

written as $c_1 \log P + c_2$, with c_1, c_2 constants independent in z, τ. Hence, the number of bits that are lost in the whole of Step 7 is

$$\sum_{i=1}^{n} \delta + h + \log(p/2^i) \leq G \log P + H$$

since the number n of steps in Newton's method is $O(\log P)$.

This means the computations in step 7 should be carried out at precision $\mathcal{P}' = \mathcal{P} + G \log P + H$, so that the result is accurate with $\mathcal{P}$ bits. This gives a running time of $O(\mathcal{M}(P) \log P)$, independently of z and τ. All the other steps cost no more than $O(\mathcal{M}(\mathcal{P}))$ bit operations. Given the formula for $\mathcal{P}$ in subsections 4.3.1 to 4.3.3, this indeed gives us a running time of $O(\mathcal{M}(P) \log P)$.

5 Implementation

An implementation using the GNU MPC library [8] for arithmetic on multiprecision complex numbers was developed; we compared our algorithm to our own implementation of Algorithm 1 using MPC[2]. The code is distributed under the GNU General public license, version 3 or any later version (GPLv3+); it is available at the address

`http://www.hlabrande.fr/pubs/fastthetas.tar.gz`

We compared those implementations to MAGMA's implementation of the computation of $\theta(z, \tau)$ (function `Theta`). Each of those implementations computed $\theta(z, \tau)$ for $z = 0.123456789 + 0.123456789i$ and $\tau = 0.23456789 + 1.23456789i$ at different precisions; the computations took place on a computer with an Intel Core i5-4570 processor. The results are presented in Figure 1 and Table 1.

Our figures show that our algorithm outperforms Magma even for computations at relatively low (i.e. 1000 digits) precision[3], and the naive algorithm for more than 325000 digits of precision. Hence, a combined algorithm which uses the naive method for precisions smaller than 325000 digits, and our method for larger precision, will yield the best algorithm, and outperform Magma in all cases, as shown in Table 1.

References

[1] J. M. Borwein and P. B. Borwein. *Pi and the AGM: a study in the analytic number theory and computational complexity*. Wiley-Interscience, 1987.

[2] J.-B. Bost and J.-F. Mestre. Moyenne arithmético-géométrique et périodes des courbes de genre 1 et 2. *Gaz. Math*, 38:36–64, 1988.

[3] R. Cosset. *Applications des fonctions thêta à la cryptographie sur courbes hyperelliptiques.* PhD thesis, Université Henri Poincaré-Nancy I, 2011.

[4] D. A. Cox. The arithmetic-geometric mean of Gauss. *Enseign. Math*, 30(2):275–330, 1984.

[5] J. E. Cremona and T. Thongjunthug. The complex AGM, periods of elliptic curves over $\mathbb{C}$ and complex elliptic logarithms. *Journal of Number Theory*, 133(8):2813–2841, August 2013.

[2] The naive algorithm which only computes $\theta(z, \tau)$ is only 5% faster; furthermore since Algorithm 5 computes all 4 values, it is fair to compare it to Algorithm 1.

[3] This is even though Magma only returns $\theta(z, \tau)$, when our algorithm returns 4 values.

[6] R. Dupont. Fast evaluation of modular functions using Newton iterations and the AGM. *Mathematics of Computation*, 80(275):1823–1847, 2011.

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

[8] A. Enge, M. Gastineau, P. Théveny, and P. Zimmerman. GNU MPC – *A library for multi-precision complex arithmetic with exact rounding*. INRIA, September 2012. Release 1.0.1, http://mpc.multiprecision.org/.

[9] A. Enge and E. Thomé. Computing class polynomials for abelian surfaces. *Experimental Mathematics*, 23(2):129–145, 2014.

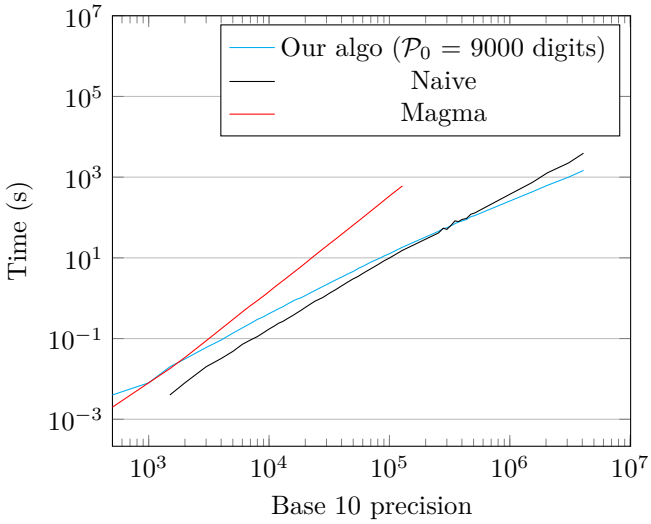
[10] H. Labrande. Absolute error in complex fixed-point arithmetic, 2015. available at `hlabrande.fr/pubs/absolutelossofprecision.pdf`.

[11] W. Luther and W. Otten. Reliable computation of elliptic functions. *Journal of Universal Computer Science*, 4(1):25–33, 1998.

[12] D. Mumford. *Tata lectures on theta*, volume I. Birkhäuser, Boston, 1983.

[13] B. Vallée. Gauss' algorithm revisited. *Journal of Algorithms*, 12(4):556–572, 1991.

Figure 1: Timing results



Prec (digits)	This work	Naive	Magma
4000	0.092	0.032	0.1740
8000	0.298	0.112	0.8719
16000	0.868	0.384	4.358
32000	2.399	1.347	22.70
64000	6.778	4.598	116.4
128000	18.32	15.29	606.1
256000	45.54	41.56	
325000	62.74	63.90	
512000	111.78	129.8	
1024000	263.7	390.3	
2048000	625.4	1275	
4096000	1468	3921	

Table 1: Times (in s) of different methods