Computing theta functions in quasi-linear time in genus 2 and above

Hugo Labrande, Emmanuel Thomé

Computing theta functions in quasi-linear time in genus 2 and above

Hugo Labrande and Emmanuel Thomé

Abstract

We outline an algorithm to compute $\theta(z, \tau)$ in genus 2 in quasi-optimal time, borrowing ideas from the algorithm for theta constants and the one for $\theta(z, \tau)$ in genus 1. Our implementation shows a large speedup for precisions as low as a few thousand decimal digits. We also lay out a strategy to generalize this algorithm to genus g.

1. *Introduction*

The θ function is important to several fields of mathematics, such as the resolution of some non-linear differential equations or the study of complex Riemann surfaces of any genus, including the important case of complex elliptic and hyperelliptic curves. The function's numerous properties underline its connection to deep topics. We refer to [**18**, **3**, **10**, **20**] and references therein for the uses of θ in various settings. Specific values of θ, called *theta constants*, are also of interest, for instance in the study of modular forms [**18**, **14**]. The problem we consider in this paper is the multiprecision computation of θ, that is to say finding a fast algorithm computing any number P of exact bits of θ. This problem has applications in the case of theta constants [**6**, **9**]; in the more general case of theta-functions, it allows to compute the Abel-Jacobi map with large precision, thereby making the algebraic–analytic link effective. Such a link offers for example an alternative way to compute isogenies using embeddings to the complex numbers.

In the case of genus 1, the theta constants exhibit a deep link with the arithmetico-geometric mean [**2**]. Using the homogeneity of the AGM gives a function which takes a simple value at the theta constants; Newton's method can then be used to compute them (see, e.g., [**5**]). This method has also been generalized in [**4**] to genus 2 theta constants, using the connection to the *Borchardt mean*; hints of a generalization to genus g are also given. Both algorithms have a *quasi-linear* asymptotic running time, i.e. they compute the first P bits of the theta constants in $O(\mathcal{M}(P) \log P)$ operations, where $\mathcal{M}(P)$ is the cost of multiplying two P-bit numbers. An implementation of the algorithm has been released in the CMH package [**8**] and used to compute class polynomials of record size [**9**].

In [**15**], we used a similar approach to design an algorithm that computes $\theta(z, \tau)$ in genus 1, for any arguments z, τ, also in asymptotic quasi-linear time. This required designing a function, inspired by the arithmetico-geometric mean, that takes a special value when evaluated at $\theta(z, \tau)$ and at the theta constants, and could be evaluated in quasi-linear time. The quasi-linear complexity beats the one of the naive algorithm, which is $O(\mathcal{M}(P)\sqrt{P})$; in practice our algorithm beats an optimized version of the naive algorithm for precision above a hundred thousand decimal digits.

In this article, we propose to generalize this strategy to theta functions of any genus. We provide a careful analysis in the case of genus 2, finding a function similar to the Borchardt mean that can also be computed with precision P in $O(\mathcal{M}(P)\log P)$, then inverting it using Newton's method. The algorithm achieves a quasi-linear complexity in P, neglecting the dependency in z and τ. We implemented this method, and provide numerical results which show that it is faster than the naive algorithm for precisions as low as 3000 decimal digits. For higher genera, we outline a way that one could generalize this algorithm in genus g, with a complexity exponential in g but quasi-linear in P. Throughout the paper, we will omit dealing with the losses in precision. We performed the full analysis in genus 1 in [**15**] and found that, given the argument reduction strategies that were set up, the loss of precision was not significant asymptotically (of the order $O(\log P)$ or $O(P)$ at the most); such a result is likely to hold, for the same reasons, in genus 2 and in genus g, but a precise analysis would be quite a lot more difficult.

This article is organized as follows. Section 2 lays out the background on genus g theta functions and algorithms to compute them; we then detail in Section 3 our algorithm for genus 2, while Section 4 shows how it could be generalized to arbitrary genus.

NOTATION 1.1. *Throughout the paper (mostly in Section 3), we use the following notation shorthand. For $(a_i)_i$ a sequence, we denote by $a_{i_0,\dots,i_{n-1}}$ the n-uple (or n-vector) $(a_{i_0},\dots,a_{i_{n-1}})$.*

2. *Background on genus g theta functions*

2.1. *Definitions*

DEFINITION 2.1 ([**18**, Section II.1]). *The* Siegel upper-half space $\mathcal{H}_g$ *is the set of symmetric $g\times g$ complex matrices whose imaginary part is positive definite.*

DEFINITION 2.2 (θ function). *Let $z\in\mathbb{C}^g$ and $\tau\in\mathcal{H}_g$. The θ function, and the associated* theta functions with characteristics *are defined as*

$$\theta(z,\tau)=\sum_{n\in\mathbb{Z}^g}\exp(i\pi\,{}^t n\tau n)\exp(2i\pi\,{}^t nz).$$

$$\text{For } a,b\in\frac{1}{2}\mathbb{Z}^g/\mathbb{Z}^g,\quad \theta_{[a;b]}(z,\tau)=\sum_{n\in\mathbb{Z}^g}\exp(i\pi\,{}^t(n+a)\tau(n+a))\exp(2i\pi\,{}^t(n+a)(z+b)).$$

Finally, theta constants *are the values in $z=0$ of the functions $\theta_{[a;b]}$.*

As done in [**4**, **1**, **9**], we will often write $\theta_{[a;b]}$ as θ_i for the integer $i=2(b_0+2b_1+\cdots+2^{g-1}b_{g-1})+2^{g+1}(a_0+2a_1+\cdots+2^{g-1}a_{g-1})$, whose binary expansion is $(2a||2b)$. We call *fundamental theta functions* the $\theta_0,\dots,\theta_{2^g-1}$, i.e. the ones with $a=0$.

PROPOSITION 2.3 (quasi-periodicity; [**18**, p. 120-123]). *For all $m \in \mathbb{Z}^g$, we have*

$$\theta_{[a;b]}(z+m,\tau) = \exp(2i\pi\,{}^t a m)\theta_{[a;b]}(z,\tau) \qquad (\theta_{[0;b]} \text{ is invariant by } z \to z+m)$$
$$\theta_{[a;b]}(z+\tau m,\tau) = \exp(-2i\pi\,{}^t b m)\exp(-i\pi\,{}^t m\tau m)\exp(-2i\pi\,{}^t m z)\theta_{[a;b]}(z,\tau)$$

2.2. *Fundamental domain*

DEFINITION 2.4. *$\mathrm{Sp}_{2g}(\mathbb{Z})$, the symplectic group of dimension g, is the set of matrices $\begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathcal{M}_{2g}(\mathbb{Z})$, such that ${}^tAC = {}^tCA$, ${}^tBD = {}^tDB$, and ${}^tAD - {}^tCB = I_g$.*

PROPOSITION 2.5 ([**14**, Prop. I.1.1]). *$\mathrm{Sp}_{2g}(\mathbb{Z})$ acts on $\mathcal{H}_g$ as follows: for $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathrm{Sp}_{2g}(\mathbb{Z})$ and $\tau \in \mathcal{H}_g$, $M \cdot \tau = (A\tau + B)(C\tau + D)^{-1} \in \mathcal{H}_g$. Furthermore, M defines an isomorphism of complex tori between $\Lambda_\tau = \mathbb{C}^g/\tau\mathbb{Z}^g + \mathbb{Z}^g$ and $\Lambda_{M\cdot\tau}$, by $z \mapsto M \cdot_\tau z = {}^t(C\tau + D)^{-1}z$; we use the shorthand $M \cdot z$ when the context allows.*

PROPOSITION 2.6 ([**14**, Def. I.3.1]). *The fundamental domain of the action of $\mathrm{Sp}_{2g}(\mathbb{Z})$ on $\mathcal{H}_g$ is the set $\mathcal{F}_g \subset \mathcal{H}_g$, defined as the matrices satisfying the conditions:*

- *$\mathrm{Im}(\tau)$ is Minkowski-reduced, i.e. ${}^tg\,\mathrm{Im}(\tau)g \geq \mathrm{Im}(\tau_{k,k})$ for all integral g with $(g_k, ..., g_n) = 1$, and $\mathrm{Im}(\tau_{k,k+1}) \geq 0$ for all k;*
- *$|\mathrm{Re}(\tau_{k,l})| \leq \frac{1}{2}$ for all $k, l \in \{1, ..., n\}, k \leq l$;*
- *$|\det(C\tau + D)| \geq 1$ for all $\begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathrm{Sp}_{2g}(\mathbb{Z})$.*

The last condition can be replaced by a finite set of inequalities; however an explicit description of those inequalities is not known in general, which means that reducing a matrix to the fundamental domain is technically not feasible. The case $g = 2$ has been solved in [**11**], which gives 19 necessary and sufficient inequalities.

THEOREM 2.7. *Let $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathrm{Sp}_{2g}(\mathbb{Z})$, and $(z, \tau) \in \mathbb{C}^g \times \mathcal{H}_g$. We have:*

$$\theta_i\left(M \cdot z, M \cdot \tau\right) = \zeta_M \sqrt{\det(C\tau + D)} e^{i\pi\,{}^t(M\cdot z)(Cz)} \theta_{\sigma_M(i)}(z, \tau) \tag{2.1}$$

where σ_M is a permutation and ζ_M is an eighth root of unity.

This theorem is proven in [**18**, Section II.5] in a special case, and in [**13**, Chap. 5, Thm. 2]; an outline of the proof can also be found in [**1**, Prop 3.1.24].

2.3. *Algorithms for theta*

A naive algorithm to compute the θ function for any genus with arbitrary precision is simply to compute the series until the remainder is small enough. This naive approach is studied for instance in [**4**, **3**]. Results giving the number of terms to compute usually require some assumptions, such as for instance $\tau \in \mathcal{F}_g$, or that the quasi-periodicity has

been used to make z small. The complexity of this method is in general $O(\mathcal{M}(P)P^{g/2})$ for P bits of precision, as we prove in Section 4.

Fast algorithms to compute theta constants in the cases $g = 1$ and $g = 2$ are known; those algorithms require $O(\mathcal{M}(P)\log P)$ operations. We expose succinctly the idea of the algorithms; more details can be found in [**5**, **4**, **9**]. The idea of both algorithms is to construct a function $\mathfrak{F}$ such that (using notation 1.1):

$$\mathfrak{F}\left(\frac{\theta_{1,\dots,2^g-1}(0,\tau)^2}{\theta_0(0,\tau)^2}\right) = \mathfrak{F}\left(\frac{\theta_1(0,\tau)^2}{\theta_0(0,\tau)^2}, \dots, \frac{\theta_{2^g-1}(0,\tau)^2}{\theta_0(0,\tau)^2}\right) = \frac{1}{\theta_0(0,\tau)^2}.$$

The function $\mathfrak{F}$ is the arithmetico-geometric mean in genus 1, and the Borchardt mean in genus 2. One then uses Equation (2.1) to find other quotients of theta constants such that evaluating $\mathfrak{F}$ at those points allows one to compute $\tau_{i,j}$. For instance, in genus 2, the following holds for τ within a large domain:

$$\mathcal{B}\left(\frac{\theta_{8,4,12}(0,\tau)^2}{\theta_0(0,\tau)^2}\right) = \mathcal{B}\left(\frac{\theta_8(0,\tau)^2}{\theta_0(0,\tau)^2}, \frac{\theta_4(0,\tau)^2}{\theta_0(0,\tau)^2}, \frac{\theta_{12}(0,\tau)^2}{\theta_0(0,\tau)^2}\right) = \frac{1}{(\tau_{12}^2 - \tau_{11}\tau_{22})\theta_0(0,\tau)^2}.$$

This also means one must be able to compute all the theta constants from the fundamental theta constants. In the end, we get a function computing τ from the quotients of fundamental theta constants; Newton's method is then applied to give an algorithm computing the theta constants.

The algorithm's complexity relies on computing $\mathfrak{F}$ efficiently, since Newton's method, when doubling the working precision at each step, does not add any extra asymptotic complexity. It is known since Gauss that the arithmetico-geometric mean converges quadratically (see e.g. [**2**]), provided one does not pick the "wrong" sign for the square root an infinite number of times; in genus 2, the Borchardt mean also converges quadratically provided similar conditions are met [**4**]. These technical requirements were shown to hold in genus 1 for τ within the fundamental domain; a similar property is conjectured to hold in genus 2 as well [**9**, Conjecture 5.7 and Remark 5.9]. In both cases, this gives a $O(\mathcal{M}(P)\log P)$ algorithm to evaluate $\mathfrak{F}$ to P bits; a modification of the algorithm can remove the dependency of the complexity in z, τ.

We successfully generalized this strategy to the computation of the genus 1 function $\theta(z,\tau)$ in [**15**]. The function $\mathfrak{F}$ is more complex, since the generalization of the AGM we consider does not converge quadratically. Instead a related sequence, obtained after considering homogenization, does. We obtained a $O(\mathcal{M}(P)\log P)$ complexity, which does not depend on z, τ. We released an GNU MPC [**7**] implementation of the algorithm, which is faster than the naive algorithm for precisions larger than 100,000 bits.

3. *Computing the genus 2 theta function*

In this section, we outline an algorithm to compute $\theta(z,\tau)$, where $z \in \mathbb{C}^2$ and $\tau \in \mathcal{H}_2$, with precision P in $O(\mathcal{M}(P)\log P)$ operations.

3.1. *Argument reduction*

For our purposes, we will not require that τ belong to the fundamental domain; weaker conditions are sufficient. Let $z = (z_1, z_2)$ and $\tau = \begin{pmatrix} \tau_1 & \tau_3 \\ \tau_3 & \tau_2 \end{pmatrix}$. We require first that τ belong to the domain $\mathcal{F}_2'$ defined by the following inequalities

$$0 \leq 2\operatorname{Im}(\tau_3) \leq \operatorname{Im}(\tau_1) \leq \operatorname{Im}(\tau_2); \quad |\operatorname{Re}(\tau_i)| \leq \frac{1}{2}; \quad \operatorname{Im}(\tau_1) \geq \sqrt{3}/2. \tag{3.1}$$

The first inequality corresponds to $\operatorname{Im}(\tau)$ being Minkowski-reduced. This domain is called $\mathcal{B}$ in [**19**], but we choose to highlight the genus in the name. We also require that z satisfy

$$|\operatorname{Re}(z_i)| \leq \frac{1}{2}; \quad |\operatorname{Im}(z_1)| \leq \frac{\operatorname{Im}(\tau_1) + \operatorname{Im}(\tau_3)}{2}; \quad |\operatorname{Im}(z_2)| \leq \frac{\operatorname{Im}(\tau_2) + \operatorname{Im}(\tau_3)}{2}. \tag{3.2}$$

Given any τ, reducing z so as to satisfy the above condition follows easily from quasi-periodicity of θ (Proposition 2.3). One can write $z = x + \tau y$, with $x, y \in \mathbb{R}^2$, explicitly by solving the system formed by $z = x + \tau y$ and $\overline{z} = x + \overline{\tau} y$; one can then subtract multiples of 1 and τ to the first argument so as to get $|x_i|, |y_i| \leq \frac{1}{2}$.

Reducing τ to the domain $\mathcal{F}_2'$ instead of $\mathcal{F}_2$ is a coarser notion, which has the advantage of being generalizable to arbitrary genus, unlike the reduction to the fundamental domain $\mathcal{F}_g$. In genus 2 the strategy for this reduction is described in [**19**, §6.3]: this gives a quasi-linear time algorithm for reducing τ to $\mathcal{F}_2'$.

3.2. *Naive algorithm*

Let $q_j = e^{i\pi\tau_j}$ and $w_j = e^{i\pi z_j}$. We have $|\theta_{[a;b]}(z,\tau)| \leq \sum_{m,n\in\mathbb{Z}} |q_1^{m^2} q_2^{n^2} q_3^{2mn} w_1^{2m} w_2^{2n}|$ using the triangular inequality. Since we have $\tau \in \mathcal{F}_2'$ and z satisfies (3.2), we have:

$$|w_1| + |w_1^{-1}| \leq 2e^{\pi\frac{3}{4}\operatorname{Im}(\tau_1)}, \qquad |w_2| + |w_2^{-1}| \leq 2e^{\pi\frac{3}{4}\operatorname{Im}(\tau_2)}.$$

Hence $(|w_1^{2m}| + |w_1^{-2m}|)(|w_2^{2n}| + |w_2^{-2n}|) \leq 4e^{i\pi(\frac{3}{2}m\operatorname{Im}(\tau_1) + \frac{3}{2}n\operatorname{Im}(\tau_2))}$. We also have

$$|q_1^{m^2} q_2^{n^2} q_3^{2mn}| \leq |q_1^{m^2} q_2^{n^2} q_3^{-m^2-n^2}| \leq |q_1^{m^2/2} q_2^{n^2/2}|$$

Hence, if S_B denotes the sum of the series defining θ with $m, n \in [-B, B]$, a calculation very similar to the one in [**15**, Prop. 2.6] proves that

$$\begin{aligned} |\theta(z,\tau) - S_B| &\leq \sum_{m\geq B} |q_1^{m^2}|(|w_1^{2m}| + |w_1^{-2m}|) + \sum_{n\geq B} |q_2^{n^2}|(|w_2^{2n}| + |w_2^{-2n}|) + 4\sum_{m,n\geq B} |q_1^{\frac{1}{2}(m-2)^2} q_2^{\frac{1}{2}(n-2)^2}| \\ &\leq \frac{4}{1-|q_1|}\left(|q_1|^{\frac{(B-2)^2}{2}-2} + |q_1|^{\frac{(B-2)^2+(B-2)^2}{2}-4}\right) \leq 5\left(|q_1|^{\frac{(B-2)^2}{2}-2} + |q_1|^{(B-2)^2+(B-2)^2-4}\right). \end{aligned}$$

This means that summing $B = O\left(\sqrt{\frac{P}{\operatorname{Im}(\tau_1)}}\right)$ terms suffices to get an approximation that is accurate to 2^{-P}.

Following and extending the strategy in [**15**, §2.2.2] or [**9**, §5.1], we use induction relations to compute terms more efficiently. Let

$$\begin{aligned} \alpha_m &= q_1^{m^2} q_2^{n^2} q_3^{2mn}(w_1^{2m} w_2^{2n} + w_1^{-2m} w_2^{-2n}), \quad \alpha'_m = q_1^{m^2} q_2^{n^2} q_3^{-2mn}(w_1^{-2m} w_2^{2n} + w_1^{2m} w_2^{-2n}), \\ \beta_n &= q_1^{m^2} q_2^{n^2} q_3^{2mn}(w_1^{2m} w_2^{2n} + w_1^{-2m} w_2^{-2n}), \quad \beta'_n = q_1^{m^2} q_2^{n^2} q_3^{-2mn}(w_1^{-2m} w_2^{2n} + w_1^{2m} w_2^{-2n}). \end{aligned}$$

Here, α_m and α'_m (respectively, β_n and β'_n) are functions of n (respectively, m). We have $\theta = \sum_{m,n\in\mathbb{N}} \alpha_m + \alpha'_m = \sum_{m,n\in\mathbb{N}} \beta_n + \beta'_n$, and

$$\begin{aligned}(w_1^2 + w_1^{-2})\alpha_{m-1} &= q_1^{-2m+1}q_3^{-2n}\alpha_m + q_1^{2m-3}q_3^{2n}\alpha_{m-2} \\ (w_1^2 + w_1^{-2})\alpha'_{m-1} &= q_1^{-2m+1}q_3^{2n}\alpha'_m + q_1^{2m-3}q_3^{-2n}\alpha'_{m-2}\end{aligned} \tag{3.3}$$

$$\begin{aligned}(w_2^2 + w_2^{-2})\beta_{n-1} &= q_2^{-2n+1}q_3^{-2m}\beta_n + q_2^{2n-3}q_3^{2m}\beta_{n-2} \\ (w_2^2 + w_2^{-2})\beta'_{n-1} &= q_2^{-2n+1}q_3^{2m}\beta'_n + q_2^{2n-3}q_3^{-2m}\beta'_{n-2}\end{aligned} \tag{3.4}$$

We propose an algorithm, Algorithm 4 (cf Appendix A), that uses those induction relations in a way such that the memory needed is only $O(1)$: it consists in iteratively computing the terms for $m = 0$ and $m = 1$ (using Equations (3.4)), and use those as soon as they are computed to initialize the other induction (Equations (3.3)). We implemented it in Magma, and will discuss timings in Section 3.6.

3.3. *The function F*

Proposition 3.1 ([**1**, formula 3.13]). *For all* $a, b \in \frac{1}{2}\mathbb{Z}^g/\mathbb{Z}^g$,

$$\theta_{[a;b]}(z,\tau)^2 = \frac{1}{2^g}\sum_{\beta\in\frac{1}{2}\mathbb{Z}^g/\mathbb{Z}^g} e^{-4i\pi\,{}^t a\beta}\theta_{[0;b+\beta]}\left(z,\frac{\tau}{2}\right)\theta_{[0;\beta]}\left(0,\frac{\tau}{2}\right). \tag{3.5}$$

We introduce the following function. Combined with Proposition 3.1, this yields Proposition 3.2.

$$\begin{aligned}F : \mathbb{C}^8 &\to \mathbb{C}^8 \\ a_{0,\dots,3}, b_{0,\dots,3} &\mapsto \left(\frac{\sqrt{a_0}\sqrt{b_0}+\sqrt{a_1}\sqrt{b_1}+\sqrt{a_2}\sqrt{b_2}+\sqrt{a_3}\sqrt{b_3}}{4}, \frac{\sqrt{a_0}\sqrt{b_1}+\sqrt{a_1}\sqrt{b_0}+\sqrt{a_2}\sqrt{b_3}+\sqrt{a_3}\sqrt{b_2}}{4},\right. \\ &\quad \frac{\sqrt{a_0}\sqrt{b_2}+\sqrt{a_1}\sqrt{b_3}+\sqrt{a_2}\sqrt{b_0}+\sqrt{a_3}\sqrt{b_1}}{4}, \frac{\sqrt{a_0}\sqrt{b_3}+\sqrt{a_1}\sqrt{b_2}+\sqrt{a_2}\sqrt{b_1}+\sqrt{a_3}\sqrt{b_0}}{4}, \\ &\quad \left.\frac{b_0+b_1+b_2+b_3}{4}, \frac{2\sqrt{b_0}\sqrt{b_1}+2\sqrt{b_2}\sqrt{b_3}}{4}, \frac{2\sqrt{b_0}\sqrt{b_2}+2\sqrt{b_1}\sqrt{b_3}}{4}, \frac{2\sqrt{b_0}\sqrt{b_3}+2\sqrt{b_1}\sqrt{b_2}}{4}\right).\end{aligned}$$

Proposition 3.2. *For a suitable choice of square roots, we have*

$$F\left(\theta_{0,1,2,3}(z,\tau)^2, \theta_{0,1,2,3}(0,\tau)^2\right) = \left(\theta_{0,1,2,3}(z,2\tau)^2, \theta_{0,1,2,3}(0,2\tau)^2\right)$$

We discuss what we mean by suitable choice of square roots. Two different notions must be considered:

– "Good choices" in the sense of [**4**, **2**], i.e. such that $\mathrm{Re}\left(\frac{\sqrt{a_i}}{\sqrt{a_j}}\right)$, $\mathrm{Re}\left(\frac{\sqrt{b_i}}{\sqrt{b_j}}\right) \geq 0$. Note that not all tuples of complex numbers admit a set of "good" square roots. In genus 1, infinitely many bad choices means the convergence is only linear; to get quadratic convergence and a quasi-linear running time, we need them to all be good after a while. This is key to our strategy to get a quasi-linear running time.

– The choice of signs that corresponds to θ, i.e. the square root such that we have $\sqrt{\theta_i(z,\tau)^2} = \theta_i(z,\tau)$. We call these the "correct" choice, which need not be a "good" choice. We need this in order to compute the right value of θ in the end.

Fortunately, the notions of "good" and "correct" choices overlap very often. In genus 1, [**2**] proves that for $z = 0$ and τ within a large domain that includes the fundamental domain, the correct choice is always good. In [**15**], we proved a similar result for arbitrary z. A reassuring fact is that in any genus, we have $\lim_{k\to\infty} \frac{\theta_{[0;b]}(z,2^k\tau)}{\theta_{[0;b']}(z,2^k\tau)} = 1$, so that the two notions coincide for large enough τ, and hence for all but a finite number of iterations of F. This argument will be key to the quadratic convergence studied in Section 3.5.

In genus 2, we do not determine an explicit domain for which correct choices are good. Although one could work with $|\theta(z,\tau) - S_B|$ to establish it, Section 3.4 would require such results for τ within a domain larger than $\mathcal{F}'_g$, making proofs difficult to obtain.

In order to overcome this difficulty and compute the correct square roots, we rely on low-precision approximations of θ – using the naive algorithm to sum the series until $\mathrm{Re}(S_B)$ or $\mathrm{Im}(S_B)$ is bigger than the bound on the norm of the remainder of the series (cf. Section 4), so that we know the sign of either $\mathrm{Re}(\theta)$ or $\mathrm{Im}(\theta)$. The number of terms and the precision needed to achieve this do not asymptotically depend on P, but only in z and τ; since we neglect the dependency in z, τ in the complexity of our algorithm[†], determining the correct square root requires only a constant number of operations. We used this strategy in our implementation; furthermore, it generalizes easily to genus g.

Proposition 3.3 computes F in 4 multiplications and 4 squares instead of the 22 multiplications its definition seems to require. Section 4 extends this to genus g.

PROPOSITION 3.3. *Let* $\left(a^{(n+1)}_{0,1,2,3}, b^{(n+1)}_{0,1,2,3}\right) = F\left(a^{(n)}_{0,1,2,3}, b^{(n)}_{0,1,2,3}\right)$. *Put* $\mathcal{H} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ *and* $\mathcal{H}_2 = \mathcal{H} \otimes \mathcal{H}$. *Let* ${}^t(m_{0,1,2,3}) = \mathcal{H}_2\, {}^t\left(\sqrt{a^{(n)}_{0,1,2,3}}\right)$ *and* ${}^t(s_{0,1,2,3}) = \mathcal{H}_2\, {}^t\left(\sqrt{b^{(n)}_{0,1,2,3}}\right)$. *We have* ${}^t(a^{(n+1)}_{0,1,2,3}) = \frac{1}{4}\mathcal{H}_2\, {}^t(m_{0,1,2,3} * s_{0,1,2,3})$ *(* $*$ *being the termwise product), and* ${}^t(b^{(n+1)}_{0,1,2,3}) = \frac{1}{4}\mathcal{H}_2\, {}^t(s^2_{0,1,2,3})$.

3.4. *Constructing and inverting the* $\mathfrak{F}$ *function*

PROPOSITION 3.4. *Define* $\mathfrak{J} = \begin{pmatrix} 0 & -I_2 \\ I_2 & 0 \end{pmatrix}$ *and* $\mathfrak{M}_i = \begin{pmatrix} I_2 & m_i \\ 0 & I_2 \end{pmatrix}$, *with* $m_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $m_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, $m_3 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ *(as in* [**4**, Chap. 6]*). Then*

$$\theta_0\left((\mathfrak{J}\mathfrak{M}_1)^2 \cdot z, (\mathfrak{J}\mathfrak{M}_1)^2 \cdot \tau\right)^2 = -\tau_1 e^{2i\pi z_1^2/\tau_1}\theta_8(z,\tau)^2,$$
$$\theta_0\left((\mathfrak{J}\mathfrak{M}_2)^2 \cdot z, (\mathfrak{J}\mathfrak{M}_2)^2 \cdot \tau\right)^2 = -\tau_2 e^{2i\pi z_2^2/\tau_2}\theta_4(z,\tau)^2,$$
$$\theta_0\left((\mathfrak{J}\mathfrak{M}_3)^2 \cdot z, (\mathfrak{J}\mathfrak{M}_3)^2 \cdot \tau\right)^2 = (\tau_3^2 - \tau_1\tau_2) e^{2i\pi \frac{z_1^2\tau_2 + z_2^2\tau_1 - 2z_1z_2\tau_3}{\det(\tau)}}\theta_0(z,\tau)^2.$$

[†]To extend the present work into an algorithm whose complexity is uniform in z, τ, one could follow the approach of [**4**, **15**].

This result is a direct consequence of Equation (2.1)[‡]. The next proposition looks at how the sequence of F^n behaves with respect to homogeneity; its proof is similar to the one in [**15**], i.e. an induction.

PROPOSITION 3.5. *Let*

$$\left({a_{0,1,2,3}}^{(n)}, {b_{0,1,2,3}}^{(n)}\right) = F^n\left(\theta_{0,1,2,3}(z,\tau)^2, \theta_{0,1,2,3}(0,\tau)^2\right),$$
$$\left({a'_{0,1,2,3}}^{(n)}, {b'_{0,1,2,3}}^{(n)}\right) = F^n\left(\lambda\theta_{0,1,2,3}(z,\tau)^2, \mu\theta_{0,1,2,3}(0,\tau)^2\right).$$

Then we have ${a'_0}^{(n)} = \epsilon_n \lambda^{1/2^n} \mu^{1-1/2^n} a_0^{(n)}$ *(where* $\epsilon_n^{2^n} = 1$*), and* ${b'_0}^{(n)} = \mu b_0^{(n)}$*, and we can compute* λ, μ *as* $\mu = \lim_{n\to\infty} b_0^{(n)}$ *and* $\lambda = \lim_{n\to\infty} \left(\frac{{a'_0}^{(n)}}{{b'_0}^{(n)}}\right)^{2^n} \times \mu$.

We define $\mathfrak{G}$ as the function which computes these two quantities; then $\mathfrak{G}\left(\lambda\theta_{0,1,2,3}(z,\tau)^2, \mu\theta_{0,1,2,3}(0,\tau)^2\right) = (\lambda,\mu)$. We prove in Section 3.5 that $\mathfrak{G}$ can be computed in $O(\mathcal{M}(P)\log P)$ operations.

We now build $\mathfrak{F}$ from $\mathfrak{G}$. The idea is to evaluate $\mathfrak{G}$ at quotients of theta functions after the action of $(\mathfrak{JM}_i)^2$; the λ and μ we recover are the inverses of the quantities in Proposition 3.4. Note that $(\mathfrak{JM}_i)^2 \cdot \tau \notin \mathcal{F}'_g$, which prevents us from generalizing proofs which worked for genus 1 to make "good choices" and "correct choices" coincide. However it is still possible to determine the sign using low-precision approximations.

Evaluating the quotients after the action of $(\mathfrak{JM}_i)^2$ is actually simply evaluating different quotients of theta functions (by Theorem 2.7); for instance:

$$\frac{\theta_1\left((\mathfrak{JM}_1)^2\cdot z, (\mathfrak{JM}_1)^2\cdot\tau\right)^2}{\theta_0\left((\mathfrak{JM}_1)^2\cdot z, (\mathfrak{JM}_1)^2\cdot\tau\right)^2} = \frac{\theta_9(z,\tau)^2}{\theta_0(z,\tau)^2}.$$

Hence, we need to compute $\theta_{[a;b]}(z,\tau)$ for $a \neq 0$. The approach used in [**4**] for theta constants is to use explicit formulas linking the $\left(\theta_{[a;b]}(0,\tau)\right)$ to the $\left(\theta_{[0;b]}(0,\tau)\right)$. Instead, we use the approach of [**9**], which is simpler and more generalizable. The τ-duplication formulas (Equation (3.5)) allow us to compute $\theta_{[a;b]}(z,2\tau)$ from the fundamental thetas, and we then compute λ and μ corresponding to the quotients at 2τ, instead of the ones corresponding to the same quotients at τ. This still gives two numbers that are a simple function of z and τ, which is all we need to use Newton's method.

Defining $\mathfrak{F}$ so that it is locally invertible, in order to use Newton's method, requires some care. In genus 1, we simply compute z and τ, which gives a $\mathbb{C}^2 \to \mathbb{C}^2$ function; however in higher genus this approach leads to a function from $\mathbb{C}^{2^{g+1}-2}$ to $\mathbb{C}^{g(g+3)/2}$, and we cannot apply Newton's method to recover the quotients of thetas. In genus 2, the function is from $\mathbb{C}^6$ to $\mathbb{C}^5$; there are two ways to work around the issue:

(1) A natural idea would be to add an extra equation, which would be the equation of a variety which the thetas lie on. For instance, we can take the equation of

[‡]Note that the result appears different from [**4**] only because the tables for $\mathfrak{M}_1$ and $\mathfrak{M}_2$ (page 146) have been switched by mistake; and it differs from [**9**] because their $\mathfrak{M}_2$ is our $\mathfrak{M}_3$, and vice-versa.

Algorithm 1 Compute $\mathfrak{F}\left(\frac{\theta^2_{1,2,3}}{\theta^2_0}(z,\tau), \frac{\theta^2_{1,2,3}}{\theta^2_0}(0,\tau)\right)$.

Evaluate $\mathfrak{G}$ at $\left(\frac{\theta^2_{1,2,3}}{\theta^2_0}(z,\tau), \frac{\theta^2_{1,2,3}}{\theta^2_0}(0,\tau)\right)$ to recover $\lambda = \frac{1}{\theta_0(z,\tau)^2}$ and $\mu = \frac{1}{\theta_0(0,\tau)^2}$.
Compute the $\theta_i(z,2\tau)^2, \theta_i(0,2\tau)^2$ using Equation (3.5).
Compute $\lambda_1, \mu_1 = \mathfrak{G}\left(\frac{\theta^2_{9,0,1}}{\theta^2_8}(z,2\tau), \frac{\theta^2_{9,0,1}}{\theta^2_8}(0,2\tau)\right)$.
Compute $\lambda_2, \mu_2 = \mathfrak{G}\left(\frac{\theta^2_{0,6,2}}{\theta^2_4}(z,2\tau), \frac{\theta^2_{0,6,2}}{\theta^2_4}(0,2\tau)\right)$.
Compute $\lambda_3, \mu_3 = \mathfrak{G}\left(\frac{\theta^2_{8,4,12}}{\theta^2_0}(z,2\tau), \frac{\theta^2_{8,4,12}}{\theta^2_0}(0,2\tau)\right)$.
$\mu_1 \leftarrow \theta_8^2(0,2\tau)/\mu_1, \mu_2 \leftarrow \theta_4(0,2\tau)^2/\mu_2, \mu_3 \leftarrow \theta_0(0,2\tau)^2/\mu_3$.
return $\left(\mu_1\theta_8^2(z,2\tau)/\lambda_1, \mu_2\theta_4^2(z,2\tau)/\lambda_2, \mu_3\theta_0(z,2\tau)^2/\lambda_3, \mu_1, \mu_2, \mu_3\right)$

the Kummer surface, as described in [**10**], which links the fundamental theta-functions and theta constants. This approach, however, does not appear to be easily generalizable to higher genus.

(2) An approach which actually works just as well is to define $\mathfrak{F}$ so that it outputs a few values of λ and μ computed by different means, instead of z, τ. In our case, we modify the λ, μ slightly, so that

$$\mathfrak{F}\left(\frac{\theta_{1,2,3}(z,\tau)^2}{\theta_0(z,\tau)^2}, \frac{\theta_{1,2,3}(0,\tau)^2}{\theta_0(0,\tau)^2}, \right) = \left(e^{i\pi\frac{z_1^2}{\tau_{11}}}, e^{i\pi\frac{z_2^2}{\tau_{22}}}, e^{i\pi\frac{z_1^2\tau_2 + z_2^2\tau_1 - 2z_1z_2\tau_3}{\det(\tau)}}, \tau_1, \tau_2, \tau_3^2 - \tau_1\tau_2\right).$$

We use here the second approach, since it can be generalized to higher genera. Our final function $\mathfrak{F}$ is thus described in Algorithm 1.

We conjecture the following, which holds experimentally:

CONJECTURE 3.6. *The Jacobian of $\mathfrak{F}$ system is invertible.*

To finish, we describe how we use Newton's method to get an approximation of θ with precision $p - \delta$, where δ is a small constant, from an approximation with precision $p/2$. We compute an approximation of $\frac{\partial \mathfrak{F}_i}{\partial x_j}$ with precision p using finite differences, i.e. $\frac{\mathfrak{F}_i(x+\epsilon_j) - \mathfrak{F}_i(x)}{||\epsilon_j||}$, for ϵ_j a perturbation of 2^{-p} on the j-th coordinate. We prove in the next section that computing $\mathfrak{F}$ with precision p costs $O(\mathcal{M}(p)\log p)$ for any arguments; this implies that applying one step of Newton's method costs $O(\mathcal{M}(p)\log p)$. Thus, as in [**4**, **9**, **15**] we can compute an approximation of θ with precision P_0 using the naive algorithm, then use Newton's method to refine it into a value of θ with precision P; the total cost of this algorithm is $O(\mathcal{M}(P)\log P)$.

3.5. *Proof of quasi-optimal time*

We prove that the computation of $\mathfrak{G}$ with precision P requires only $O(\log P)$ steps as long as the choice of signs is always good. The result if the arguments are $\left(\frac{\theta_1^2}{\theta_0^2}, \frac{\theta_2^2}{\theta_0^2}, \frac{\theta_3^2}{\theta_0^2}\right)$ is merely a consequence of the quadratic convergence of $(\theta(z, 2^k\tau))_{k\in\mathbb{N}}$; however we need to prove the result for any arguments to apply it to the computation of the Jacobian.

PROPOSITION 3.7. *Suppose that the choice of signs is always good. Then*
(i) *the* $|a_0^{(i)}|, |b_0^{(i)}|$ *are upper-bounded;*
(ii) *the* $|a_0^{(i)}|, |b_0^{(i)}|$ *are lower-bounded;*
(iii) *only* $O(\log P)$ *steps are needed to compute* λ *with precision* P.

Proof. The proof is very similar to [**15**, Section 3.4], hence we omit some details.
(i) Easy induction using the definition of F.
(ii) The $|b_i^{(n)}|$ are lower bounded by [**4**, Prop. 7.3]. We follow the proof of [**15**, Prop. 3.8]. The $|a_i - a_j|$ converge quadratically since the choice of sign is good:

$$|a_0^{(n+1)} - a_1^{(n+1)}|^2 = \frac{|m_2 + m_3|^2}{4} \qquad \text{(using notations of Prop 3.3)}$$
$$\leq 2C|\sqrt{b_0^{(n)}} - \sqrt{b_1^{(n)}}|^2 \leq 2C|b_0^{(n)} - b_1^{(n)}|.$$

We can also prove inequalities of the shape $|\sqrt{a_i} + \sqrt{a_j}| \geq c_1\sqrt{1 - |a_i - a_j|/c_2}$ using the parallelogram identity. Those inequalities can be combined to get a lower bound for $|a_i^{(n+1)}|$ that proves that if $|a_i^{(n)}|, |b_i^{(n)}| \geq \delta$, then $|a_i^{(n+1)}|, |b_i^{(n+1)}| \geq \delta(1 + \epsilon_n)$ with ϵ_n quadratically convergent. This finishes the proof.
(iii) Proceed exactly as in [**15**, Prop. 3.9]; the only difference is

$$a_0^{(n+1)} - \sqrt{a_0^{(n)} b_0^{(n)}} = \frac{\sum_{i=1}^{3} (\sqrt{a_i^{(n)}} - \sqrt{a_0^{(n)}})(\sqrt{b_i^{(n)}} + \sqrt{b_0^{(n)}}) + (\sqrt{a_i^{(n)}} + \sqrt{a_0^{(n)}})(\sqrt{b_i^{(n)}} - \sqrt{b_0^{(n)}})}{8}$$
$$\leq \frac{\sqrt{C}}{4} \sum_{i=1}^{3} |\sqrt{a_i^{(n)}} - \sqrt{a_0^{(n)}}| + |\sqrt{b_i^{(n)}} - \sqrt{b_0^{(n)}}|$$
$$\leq \frac{\sqrt{C}}{4} \sum_{i=1}^{3} \sqrt{|a_i^{(n)} - a_0^{(n)}|} + \sqrt{|b_i^{(n)} - b_0^{(n)}|}.$$

Then use the calculation in (ii) to prove quadratic convergence; the quadratic convergence of q_n to 1, and hence the result, follows exactly the same. □

3.6. *Implementation results*

We wrote an implementation of this quasi-optimal time algorithm in Magma and compared it with Magma's `Theta` function. In addition, we implemented Algorithm 4, i.e. the naive algorithm combined with induction relations, achieving a $O(\mathcal{M}(P)P)$ complexity. Magma's `Theta` function is probably general-purpose code, which means it probably uses exponentiations to compute each term, making it a $O(\mathcal{M}(P)P \log P)$ algorithm at best; however its complexity in practice appears to be much worse. Our implementation of the quasi-optimal time algorithm uses our naive algorithm to compute a first approximation of the values of θ with 3000 digits, which is the crossover point for which our algorithm is better than the naive algorithm.

Our results in Table 1 show that for precisions higher than 1000 decimal digits our algorithm, which outputs 8 values, is faster than one call to Magma's `Theta` function, which simply computes one value of $\theta(z, \tau)$. Furthermore, our quasi-optimal algorithm is better than Algorithm 4 for precisions greater than 3000 digits; this is much earlier than in the genus 1 case, which is explained by the fact that the complexity of the

Algorithm 2 Reduce τ. **Input:** $\tau \in \mathcal{H}_g$ **Output:** $\tau' \in \mathcal{F}'_g$.

1: $\tau' \leftarrow \tau$.
2: Apply Minkowski reduction to τ'.
3: Subtract an integer matrix to τ' so that $|\operatorname{Re}(\tau'_{i,j})| \leq \frac{1}{2}$.
4: If $|\tau'_{1,1}| \leq 1$, do $\tau' \leftarrow N_0 \cdot \tau'$ and go back to Step 2.
5: **return** τ'.

naive algorithm is $O(\mathcal{M}(P)\sqrt{P})$ in genus 1 and $O(\mathcal{M}(P)P)$ in genus 2. Our results are consistent with the situation in the case of theta constants, studied in [**9**]†.

Prec (digits)	Magma	Algorithm 4	This work
1000	0.42	0.38	0.38
2000	2.58	1.86	1.86
4000	18.4	9.51	6.65
8000	128.98	53.85	13.17
16000	889	303	25
32000	6368	1535	50
64000	46566	8798	120

Table 1: Times (in s) of different methods

4. *Extending the algorithm to higher genera*

This section outlines ideas for extending the previous strategy to the case $g > 2$. The complexity of such an algorithm will certainly be exponential (or worse) in g; we do not make any attempt at lowering this complexity, and in fact we do not even evaluate it fully. However, the complexity in P would still be $O(\mathcal{M}(P)\log P)$, which is desirable.

4.1. *Argument reduction*

We extend the domain $\mathcal{F}'_2$ to genus g as follows: $\mathcal{F}'_g$ is the set of τ such that:

$$\operatorname{Re}(\tau_{i,j}) \leq \frac{1}{2}; \qquad \operatorname{Im}(\tau) \text{ is Minkowski-reduced}; \qquad \operatorname{Im}(\tau_{1,1}) \geq \sqrt{3}/2.$$

Note that $\mathcal{F}_g \subset \mathcal{F}'_g$, since $N_0 = \begin{pmatrix} I_g - \delta_{1,1} & -\delta_{1,1} \\ \delta_{1,1} & I_g - \delta_{1,1} \end{pmatrix}$, where $\delta_{1,1}$ is the $g \times g$ Kronecker matrix, is symplectic and such that $|\det(C\tau + D)| = |\tau_{1,1}|$. The algorithm to reduce τ is similar to [**19**, Alg. 6.8]; this is Algorithm 2.

Proposition 4.1. *Algorithm 2 terminates.*

†Note that our code is slower by several orders of magnitude; however, their algorithm only computes theta constants, and their implementation is written in low-level C (MPC) while ours uses Magma. An MPC implementation of our algorithm would speed up the computations significantly.

Proof. We generalize the lemmas in [**19**, Section 6.4]; the proof is rather technical. Lemma 6.9 holds in genus g; Lemma 6.14 becomes

$$(m_2 \cdots m_g)(\mathrm{Im}(M(Z))) \leq c_1(g) \max\{m_1(Y)^{-1}, (m_2 \cdots m_g)(Y)\}$$

with $c_1(g)$ the constant in Minkowski's inequality [**14**, Prop I.2.1, p.13].

We generalize Lemma 6.12 as follows. Let R_g be the set of Minkowski-reduced matrices, and $Q'_g(t)$ defined as in [**14**, Def. I.2.3]; then, the remark after [**14**, Def. I.2.2] and [**14**, Prop. I.2.2] proves that there is a $t' > 2$ such that $R_g \subset Q'_g(t')$. Consider the set of the matrices τ' obtained during the execution of Algorithm 2 for which $y_1 \geq \frac{1}{t'}$; this set injects in the set $L_g(t')$ defined in [**14**, Def. I.3.2]. Applying [**14**, Thm. I.3.1] yields the result that there are only finitely many steps in which $y_1 > \frac{1}{t'}$; note c the number of such steps (i.e. the cardinality of the set in [**14**, Thm. I.3.1]).

Lemma 6.11 with the bound $y_1 \leq \frac{1}{t'}$ holds in genus g, since $t' > 2$. Combining all the lemmas as in [**19**, Prop. 6.13] proves termination, since after k iterations

$$2^{k-c} \leq c_1(g)^3 \max\{m_1(Y_0)^{-3}(m_2 \cdots m_g)(Y_0)^{-1}, (m_2 \cdots m_g)(Y_0)m_1(Y_0)^{-1}\} \qquad \square$$

Bounding the number of steps in the algorithm requires making a few theorems explicit, namely [**14**, Prop. I.2.2] (making t' explicit) and [**14**, Thm. I.3.1] (determining c). We believe that the number of steps is exponential in g. Furthermore, each step requires computing the Minkowksi reduction of a $g \times g$ matrix, which has a cost of $O\left(2^{1.3g^3}\right)$ arithmetic operations [**12**]. Hence the running time of this reduction is exponential in g.

The conditions on z, which can be met using Proposition 2.3, are

$$|\mathrm{Re}(z_i)| \leq \frac{1}{2}; \qquad |\mathrm{Im}(z_i)| \leq \frac{\sum_{j \in [1..2g]} |\mathrm{Im}(\tau_{i,j})|}{2}.$$

We note that [**3**] uses another approach, the so-called Siegel reduction, with weaker conditions than the ones we impose here, e.g. using LLL reduction instead of Minkowski reduction. It is apparently enough to limit the number of terms in the naive algorithm.

4.2. *Naive algorithm*

In [**3**], the authors compute an ellipsoid containing the indices of the terms one needs to sum to get an approximation of $\theta(z, \tau)$ up to ϵ. Its size depends on R, defined as the solution to the equation $\epsilon = \frac{g}{2} \frac{2^g}{\rho^g} \Gamma(g/2, (R - \rho/2)^2)$, where Γ is the incomplete gamma function and ρ the smallest vector after an orthogonal change of basis.

Neglecting the dependency in τ and z, we get the rather coarse bound of $O(R^g)$ terms needed. We complete the analysis in [**3**] by computing an explicit estimate on R:

PROPOSITION 4.2. *Treating z, τ (and hence ρ) as constants, we have $R = O(\sqrt{P})$, i.e. summing $O(P^{g/2})$ terms is sufficient to get a result accurate to P bits.*

Proof. Assuming that g is even (which we can do since Γ is growing in the first parameter for R large enough), we use integration by parts $g/2$ times to prove that

$$\begin{aligned}\Gamma(g/2,d) &= (g/2-1)!e^{-d} + \sum_{i=1}^{g/2}(g/2-1)\cdots(g/2-i)d^{g/2-i}e^{-d}\\ &\leq \frac{g}{2}(g/2-1)!d^{g/2-1}e^{-d} \leq e^{-d+g/2(\log d+\log(g/2))}.\end{aligned}$$

$$\text{Hence :}\quad \frac{g}{2}\frac{2^g}{\rho^g}\Gamma(g/2,d) \leq 2^{-d\log_2 e+g/2(\log d+\log(g/2))+\log(g/2)+g\log(2/\rho)}.$$

Asymptotically, i.e. for R large enough, taking $d = P\log_2 e + g\log P + g\log(2/\rho) + g = O(P)$ is enough for the right hand side to be smaller than 2^{-P}. Hence $R = O(\sqrt{P})$. □

The terms can be computed using induction relations, which exist whatever the genus is: if $g-1$ indices are fixed, there exists a relation between three consecutive terms for the remaining index. However, exploiting those relations gets increasingly complicated and harder to code efficiently as the genus grows. We assume that such induction relations are used in the naive algorithm, i.e. that the cost of computing each term is only $O(\mathcal{M}(P))$, and the memory needs $O(1)$ or $O(g)$. Under this assumption, the cost of the naive algorithm is $O(\mathcal{M}(P)P^{g/2})$ operations, which agrees with our analyses in genus 1 and 2.

4.3. *The function F*

We use once again the τ-duplication formula (Equation (3.5)) where $a = 0$:

$$\theta_i(z,2\tau)^2 = \frac{1}{2^g}\sum_{k\in\{0,\ldots,2^g-1\}}\theta_{i\oplus k}(z,\tau)\theta_k(0,\tau), \tag{4.1}$$

where $\oplus$ is the bitwise XOR. This gives us a function

$$\begin{gathered}F : \mathbb{C}^{2^{g+1}} \to \mathbb{C}^{2^{g+1}}\\ \left(\left(\theta^2_{[0;b]}(z,\tau)\right),\left(\theta^2_{[0;b]}(0,\tau)\right)\right) \mapsto \left(\left(\theta^2_{[0;b]}(z,2\tau)\right),\left(\theta^2_{[0;b]}(0,2\tau)\right)\right).\end{gathered}$$

Just like in genus 2, we solve the problem of determining the correct square root by using low-precision approximations of θ, which only require a number of terms and a precision that are independent of P.

The trick we used in Proposition 3.3 can be generalized here. Sums involving bitwise XORs as in Equation (3.5) are called *dyadic convolutions* in [**17**], who also gives an optimal algorithm to compute them. The method is exactly the one we used in Proposition 3.3, using this time $\mathcal{H}_g = \mathcal{H}\otimes\cdots\otimes\mathcal{H}$ (g times). This means we only need 2^{g+1} multiplications, instead of the 2^{2g+1} multiplications in the definition of F.

4.4. *Extending the quasi-linear time algorithm*

4.4.1. *Defining $\mathfrak{F}$.* We can study the homogeneity of the formulas defining F; Equation (3.5) gives for instance $\theta(z,2\tau)^2 = \sum_{i=0}^{2^g-1}\sqrt{\theta_i(z,\tau)^2}\sqrt{\theta_i(0,\tau)^2}$. Combined

Algorithm 3 Compute $\mathfrak{F}\left(\frac{\theta_{1,\dots,2^g-1}(z,\tau)^2}{\theta_0(z,\tau)^2}, \frac{\theta_{1,\dots,2^g-1}(0,\tau)^2}{\theta_0(0,\tau)^2}\right)$.

1: Compute $\lambda_0, \mu_0 = \frac{1}{\theta_0(z,\tau)^2}, \frac{1}{\theta_0(0,\tau)^2} = F^\infty\left(\frac{\theta_{1,\dots,2^g-1}(z,\tau)^2}{\theta_0(z,\tau)^2}, \frac{\theta_{1,\dots,2^g-1}(0,\tau)^2}{\theta_0(0,\tau)^2}\right)$.
2: Compute the individual $\theta_i(z,\tau), \theta_i(z,\tau)$ for $i \in \{0,...,2^g-1\}$.
3: Use Equation (3.5) to compute $\theta_i(z,2\tau)^2, \theta_i(0,2\tau)^2$ for $i \in \{0,...,2^{2g}-1\}$.
4: **for** i = 1 to $2^g - 1$ **do**
5: Compute $\lambda_i, \mu_i = F^\infty\left(\frac{\theta_{1,\dots,2^g-1}(\mathfrak{M}_i\cdot z, \mathfrak{M}_i\cdot 2\tau)^2}{\theta_0(\mathfrak{M}_i\cdot z, \mathfrak{M}_i\cdot 2\tau)^2}, \frac{\theta_{1,\dots,2^g-1}(0, \mathfrak{M}_i\cdot 2\tau)^2}{\theta_0(0, \mathfrak{M}_i\cdot 2\tau)^2}\right)$.
6: **end for**
7: **return** (λ_i, μ_i).

with the expression of the Borchardt mean, it is then easy to prove that

$$\mu = \lim_{n\to\infty} b_0^{(n)}, \qquad \lambda = \left(\frac{a_0^{(n)}}{b_0^{(n)}}\right)^{2^n} \times \mu.$$

We then consider the sequence formed of iterates of F. The Borchardt mean of 2^g numbers converges quadratically [**4**, Chap. 7] (modulo some rather weak hypotheses) . We put forward the following conjecture, which has been proven in genus 1 and 2:

Conjecture 4.3. *If all the choices of sign are good, λ and μ can be computed up to 2^{-P} in $O(\log P)$ steps.*

This means that F^∞, the function computing λ, μ with precision P from the quotients of theta functions and theta constants, can be evaluated in $O(2^g \mathcal{M}(P) \log P)$ operations.

The other requirement for defining $\mathfrak{F}$ in a way that Newton's method is applicable is to find $2^g - 1$ symplectic matrices M_i such that

$$\theta_0(M_i \cdot z, M_i \cdot \tau)^2 = f(\tau) e^{2i\pi g(z,\tau)} \theta_{n_i}(z,\tau)^2$$

with f, g rational functions. We then define the $\mathfrak{F}$ function with Algorithm 3.

4.4.2. *The final algorithm.* We can compute λ_i, μ_i from z, τ using Equation 2.1 in $O(2^g \mathcal{M}(P) \log P)$ operations. We can then apply Newton's method to $\mathfrak{F}$ to compute the quotients of theta functions and theta constants, but only if the Jacobian of the system is invertible. This depends on which M_i are chosen, but we conjecture that such a choice can be found, as in genus 1 and 2.

The total complexity of this method is the same as the complexity of evaluating $\mathfrak{F}$, since Newton's method (when doubling the working precision at each step) does not add any asymptotic complexity. The complexity of the evaluation of $\mathfrak{F}$ is $O(4^g \mathcal{M}(P) \log P)$ bit operations. Although this is exponential in the genus g, this is quasi-linear in the precision P; hence, as it was the case between genus 1 and 2, we expect the precision for which our algorithm is better than a naive approach to be smaller as the genus grows.

References

1. R. Cosset. *Applications des fonctions thêta à la cryptographie sur courbes hyperelliptiques*. PhD thesis, Université Henri Poincaré-Nancy I, 2011.
2. D. A. Cox. The arithmetic-geometric mean of Gauss. *Enseign. Math*, 30(2):275–330, 1984.
3. B. Deconinck, M. Heil, A. Bobenko, M. Van Hoeij, and M. Schmies. Computing Riemann theta functions. *Math. Comp.*, 73(247):1417–1442, 2004.
4. R. Dupont. *Moyenne arithmético-géométrique, suites de Borchardt et applications*. PhD thesis, École polytechnique, Palaiseau, 2006. `http://www.lix.polytechnique.fr/Labo/Regis.Dupont/these_soutenance.pdf`.
5. R. Dupont. Fast evaluation of modular functions using Newton iterations and the AGM. *Math. Comp.*, 80(275):1823–1847, 2011.
6. A. Enge. The complexity of class polynomial computation via floating point approximations. *Math. Comp.*, 78(266):1089–1107, 2009.
7. A. Enge, M. Gastineau, P. Théveny, and P. Zimmerman. GNU MPC. INRIA, September 2012. Release 1.0.1, `http://mpc.multiprecision.org/`.
8. A. Enge and E. Thomé. *CMH — Computation of Igusa Class Polynomials*, Dec. 2014. Version 1.0, `http://cmh.gforge.inria.fr/`.
9. A. Enge and E. Thomé. Computing class polynomials for abelian surfaces. *Exp. Math.*, 23(2):129–145, 2014.
10. P. Gaudry. Fast genus 2 arithmetic based on theta functions. *J. Math. Cryptol.*, 1(3):243–265, 2007.
11. E. Gottschling. Explizite bestimmung der randflächen des fundamentalbereiches der modulgruppe zweiten grades. *Math. Ann.*, 138(2):103–124, 1959.
12. B. Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41:125–139, 1985.
13. J.-I. Igusa. *Theta functions*. Springer, 1972.
14. H. Klingen. *Introductory lectures on Siegel modular forms*. Cambridge University Press, 1990.
15. H. Labrande. Computing Jacobi's θ in quasi-linear time. `http://arxiv.org/abs/1511.04248`, 2015.
16. W. Luther and W. Otten. Reliable computation of elliptic functions. *J.UCS*, 4(1):25–33, 1998.
17. O. Makarov. The connection between algorithms of the fast Fourier and Hadamard transformations and the algorithms of Karatsuba, Strassen, and Winograd. *USSR Computational Mathematics and Mathematical Physics*, 15(5):1–11, 1975.
18. D. Mumford. *Tata lectures on Theta*, volume I. Birkhäuser, Boston, 1983.
19. M. Streng. Computing Igusa class polynomials. *Math. Comp.*, 83(285), 2014.
20. P. Van Wamelen. Equations for the Jacobian of a hyperelliptic curve. *Trans. Amer. Math. Soc.*, 350(8):3083–3106, 1998.

Hugo Labrande
Université de Lorraine, LORIA (UMR CNRS 7503), INRIA Nancy (CARAMBA) & University of Calgary
615 rue du jardin botanique,
54602 Villers-lès-Nancy Cedex, France

hugo.labrande@inria.fr

Emmanuel Thomé
INRIA Nancy (CARAMBA), LORIA (UMR CNRS 7503), Université de Lorraine
615 rue du jardin botanique,
54602 Villers-lès-Nancy Cedex, France

emmanuel.thome@inria.fr

Appendix A. *Naive algorithm in genus 2*

Algorithm 4 computes $\theta(z, \tau)$ in genus 2 using the partial evaluation of the series, as well as induction relations to speed up the computation of each term. The terms of the form q_i^k should be computed using induction relations; we used such relations in our implementation, but did not include this here for readability. There are ways to speed up this algorithm using $O(\sqrt{P})$ memory, for instance by caching the q_1^m and the q_2^n; we did not investigate those methods.

Algorithm 4 Naive algorithm for $\theta(z,\tau)$ in genus 2.
Input: z,τ Output: $\theta_i(z,\tau),\theta_i(0,\tau)$ for $i\in\{0..3\}$.

For $i := 1$ to 4, tz[i] $\leftarrow 1$, t0[i] $\leftarrow 1$ ▷ We start with the sums for $n=0$.
$u_{0,0}\leftarrow 1, u_{1,0}\leftarrow q_1$, $v_{0,0}\leftarrow 1, v_{1,0}\leftarrow q_1(w_1^2+w_1^{-2})$. Add those to the tz[i], t0[i] with the correct sign.
for m = 2 to B_1 **do**
    $u_{m,0}\leftarrow q_1^{2m+1}u_{m-1,0}$. Add to the t0[i] with the correct sign.
    $v_{m,0}\leftarrow q_1^{2m-1}v_{m-1,0}v_1 - q_1^{4m-4}v_{m-2,0}$. Add to the tz[i] with the correct sign.
end for ▷ Now for the rest of the sum
$v_0\leftarrow 2, v_1\leftarrow q_3^2+q_3^{-2}$; $w_0^{m=0}\leftarrow 2, w_0^{m=1}\leftarrow q_1q_2(q_3^2+q_3^{-2})$.
$\beta_0^{m=0}\leftarrow 2, \beta_0^{m=1}\leftarrow q_1(w_1^2+w_1^{-2}), \beta_1^{m=0}\leftarrow q_2(w_2^2+w_2^{-2}), \beta_1^{m=1}\leftarrow q_1q_2q_3^2(w_1^2w_2^2+w_1^{-2}w_2^{-2})$.
$\beta_0'^{m=0}\leftarrow \beta_0^{m=0}, \beta_0'^{m=1}\leftarrow \beta_0^{m=1}, \beta_1'^{m=0}\leftarrow \beta_1^{m=0}, \beta_1'^{m=1}\leftarrow q_1q_2q_3^{-2}(w_1^{-2}w_2^2+w_1^2w_2^{-2})$.
for n = 1 to B **do**
    Add $w_0^{m=1}$ to the t0[i] and $\beta_i^{m=1}+\beta_i'^{m=1}$ to the tz[i], with the correct sign.
    $\alpha_0\leftarrow \beta_1^{m=0}$, $\alpha_1\leftarrow \beta_1^{m=1}$
    $\alpha_0'\leftarrow \beta_1'^{m=0}$, $\alpha_1'\leftarrow \beta_1'^{m=1}$
    for m=2 to $B-n^2\operatorname{Im}(\tau_2)$ **do**
        $w_m\leftarrow q_1^{2m-1}v_nw_{m-1}-q_1^{4m-4}w_{m-2}$.
        $\alpha_m\leftarrow (w_1^2+w_1^{-2})q_1^{2m-1}q_3^{2n}\alpha_{m-1}-q_3^{4n}q_1^{4m-4}\alpha_{m-2}$
        $\alpha_m'\leftarrow (w_1^2+w_1^{-2})q_1^{2m-1}q_3^{-2n}\alpha_{m-1}'-q_3^{-4n}q_1^{4m-4}\alpha_{m-2}'$
        Add w_m to the t0[i] and $\alpha_m+\alpha_m'$ to the tz[i], with the correct sign.
    end for
    $v_{n+1}\leftarrow v_nv_1-v_{n-1}$
end for
return tz, t0.