Evaluating modular polynomials in genus 2

Jean Kieffer

Evaluating modular polynomials in genus 2

Jean Kieffer

October 19, 2020

Abstract

We design algorithms to efficiently evaluate genus 2 modular polynomials of Siegel and Hilbert type over number fields, using complex approximations. Under heuristics related to the computation of theta functions in quasi-linear time, the output is provably correct. Our algorithms also apply to finite fields via lifting.

1 Introduction

The Siegel and Hilbert modular polynomials are genus 2 analogues of the classical, genus 1 modular polynomials. If $\ell \in \mathbb{Z}$ is a prime, the Siegel modular polynomials of level ℓ, denoted by $\Psi_{\ell,1}, \Psi_{\ell,2}, \Psi_{\ell,3} \in \mathbb{Q}(j_1, j_2, j_3)[X]$, encode the presence of ℓ-isogenies between Jacobians of genus 2 curves: if $\mathcal{C}$ and $\mathcal{C}'$ are genus 2 curves over a field k with Igusa invariants (j_1, j_2, j_3) and (j_1', j_2', j_3') respectively, and if no division by zero occurs, then the equalities

$$\Psi_{\ell,1}(j_1, j_2, j_3, j_1') = 0 \quad \text{and} \quad j_k' = \frac{\Psi_{\ell,k}(j_1, j_2, j_3, j_1')}{\partial_X \Psi_{\ell,1}(j_1, j_2, j_3, j_1')} \quad (k = 2, 3)$$

hold if and only if the Jacobians $\mathrm{Jac}(\mathcal{C})$ and $\mathrm{Jac}(\mathcal{C}')$ are ℓ-isogenous over an algebraic closure of k [14]. Similarly, Hilbert modular polynomials describe certain cyclic isogenies between Jacobians with fixed real multiplication [12, 15].

From a computational point of view, modular equations are useful to detect isogenies without prior knowledge of their kernels, and also to compute these isogenies explicitly [11]. The drawback is that genus 2 modular equations, of Siegel type in particular, are very large objects. The degree of the polynomials $\Psi_{\ell,k}$ in each variable is $O(\ell^3)$, and the height of their coefficients is $O(\ell^3 \log \ell)$ [9]. Therefore, merely storing these polynomials costs $O(\ell^{15} \log \ell)$ space.

In this paper, we argue that precomputing modular equations in full is not the correct strategy in higher genus. Indeed, in most contexts, we only need *evaluations* of modular equations, and possibly their derivatives, at a given triple $(j_1, j_2, j_3) \in F^3$, where F is a number field; finite fields reduce to this case via lifts. Computing these univariate polynomials directly is more much efficient: we prove the following theorem under heuristics related to the computation of theta functions in quasi-linear time.

Theorem 1.1 (Under hypothesis 4.1)**.** *There exists an algorithm which, given prime numbers p and ℓ, and given $(j_1, j_2, j_3) \in \mathbb{F}_p^3$ where the denominators of the Siegel modular polynomials $\Psi_{\ell,k}$ do not vanish, evaluates $\Psi_{\ell,k}(j_1, j_2, j_3) \in \mathbb{F}_p[X]$ for $1 \leq k \leq 3$ within $\widetilde{O}(\ell^3 \log^2 p + \ell^6 \log p)$ binary operations.*

A similar results holds for Hilbert modular polynomials, with a complexity of only $\widetilde{O}(\ell \log^2 p + \ell^2 \log p)$ binary operations. In both cases, we save a factor of $\log p$ when the input values can be written as quotients of small integers.

Overview of the algorithm. Let $\mathcal{C}$ be a genus 2 hyperelliptic curve over F with Igusa invariants (j_1, j_2, j_3). In every complex embedding of F, we compute a period matrix of $\mathcal{C}$ in the Siegel half space $\mathcal{H}_2$. Then, we compute approximations of the numerator and denominator of $\Psi_{\ell,k}(j_1, j_2, j_3)$ using analytic formulas: this is done by computing theta constants at all period matrices representing Jacobians ℓ-isogenous to $\mathrm{Jac}(\mathcal{C})$. At the end, we recognize the coefficients as algebraic integers. During the algorithm, we keep track of precision losses in order to obtain a provably correct result.

Computational model. We use interval arithmetic: given $z \in \mathbb{C}$, an *approximation of z to precision N* is a complex ball centered in some $z' \in \mathbb{C}$ of radius 2^{-N} containing z. An approximation of a polynomial to precision N is an approximation to precision N coefficient per coefficient. In this model, precision losses can be predicted in terms of the size of the operands, and can also be computed on the fly in a precise way: this is done in the Arb library [8]. In the eventuality that our algorithm runs out of precision during the computation, we simply double the precision and restart: therefore we only give precision losses in the big-O notation, which is certainly less cumbersome than computing constants by hand.

Organization of the paper. In Section 2, we study precision losses in operations on polynomials appearing in the sequel, namely reconstructing polynomials from their roots, Lagrange interpolation, and the reconstruction of integers in number fields from their values in complex embeddings. In Section 3, we recall the definition of Siegel and Hilbert modular polynomials and give explicit formulæ for their denominators. In Section 4, we study the critical part of the algorithm in terms of running time, namely the computation of theta constants. For a *fixed* τ lying in the Siegel fundamental domain $\mathcal{F}_2$, a heuristic algorithm by Dupont [5] computes theta functions at τ in quasi-linear time in the precision; we make the hypothesis (4.1) that this algorithm is correct and has a uniform quasi-linear time complexity in a compact subset of $\mathcal{F}_2$. Then we describe an algorithm to compute theta constants at a given $\tau \in \mathcal{F}_2$ with uniform quasi-linear cost, and analyze the precision losses when reducing other inputs to $\mathcal{F}_2$. In Section 5, we bound the cost of computing the period matrix of a given algebraic curve in terms of the height of its coefficients. Finally, we detail the evaluation algorithm in the case of Hilbert modular polynomials in Section 6.

2 Precision losses in operations on polynomials

If $\mathcal{A}$ is an algorithm taking approximations to precision N as input, we say that the precision loss in $\mathcal{A}$ is M bits if the output of $\mathcal{A}$ is an approximation of the theoretical output to precision $N - M$. We let $\mathcal{M}(N)$ be a quasi-linear, superlinear function such that two N-bit integers can be multiplied in $\mathcal{M}(N)$ binary operations. We write log for the logarithm in base 2, and for $x \in \mathbb{R}$, we define

$$\log^+ x = \log \max\{1, x\}.$$

We denote the modulus of the largest coefficient in a polynomial P by $|P|$; we also use this notation for vectors and matrices.

2.1 Elementary operations

To be short, additions can be done in linear time with a precision loss of $O(1)$, and multiplications, inversions, and square roots can be done in quasi-linear time with a precision loss given by the size of the input. We state these standard facts without proof.

Lemma 2.1. *Let $z \in \mathbb{C}^\times$ and $N \geq -\log|z| + 1$.*

1. *Given an approximation of z to precision N, the inverse $1/z$ can be computed within $O(\mathcal{M}(N + \log^+|z|))$ binary operations, with a precision loss of $-2\log|z| + O(1)$ bits.*
2. *Given an approximation of z to precision N, an approximation of a square root of z can be computed within $O(\mathcal{M}(N + \log^+|z|))$ binary operations, with a precision loss of $-\frac{1}{2}\log|z| + O(1)$ bits.*

In Lemma 2.1, the assumption on N ensures that the interval approximating z does not contain zero.

Lemma 2.2. *Let $P_1, P_2 \in \mathbb{C}[X]$, and $N, N_1, N_2 \geq 1$. Assume that P_1, P_2 and their approximations have degree at most d.*

1. *Given approximations of P_1, P_2 to precision N, the sum $P_1 + P_2$ can be computed within*

$$O\big((d+1)(N + \log\max\{1, |P_1|, |P_2|\})\big)$$

binary operations, with a precision loss of $O(1)$ bits.

2. *Given approximations of P_i to precision N_i for $i = 1, 2$, the product P_1P_2 can be computed within*

$$O\big(\mathcal{M}\big((d+1)\max\{N_1 + \log|P_1|, N_2 + \log|P_2|\}\big)\big)$$

binary operations, to precision

$$\min\{N_1 - \log^+|P_2|, N_2 - \log^+|P_1|\} - \log(1+d) - O(1).$$

2.2 Reconstruction from the roots, interpolation

Lemma 2.3. *There exists an algorithm such that the following holds. Let $d \geq 1$, $B \geq 1$, $C \geq 1$, and let x_i, y_i, z_i for $1 \leq i \leq d$ be complex numbers such that*

$$\log^+ |x_i| \leq B, \quad \log^+ |y_i| \leq B, \quad \log^+ |z_i| \leq C, \quad \text{for all } i.$$

Let $N \geq 1$. Then, given approximations of these complex numbers to precision N, the algorithm computes the polynomials

$$P = \prod_{i=1}^{d} (x_i X + y_i) \qquad Q = \sum_{i=1}^{d} z_i \prod_{j \neq i} (x_j X + y_j)$$

within $O\big(\mathcal{M}\big(d(N + C + dB)\big) \log d\big)$ binary operations, with a precision loss of $O(C + dB)$ bits.

Proof. We use product trees [3, §I.5.4]. For each $0 \leq m \leq \lceil \log_2(d) \rceil$, the m-th level of the product tree for P consists in $2^{\lceil \log_2(d) \rceil - m}$ products of (at most) 2^m factors of the form $x_i X + y_i$. Hence, for every polynomial R appearing at the m-th level, we have

$$\deg(R) \leq 2^m, \quad \log^+ |R| = O(2^m B).$$

Level 0 is given as input. In order to compute leve $m + 1$ from level m, we compute one product per vertex, for a total cost of $O\big(\mathcal{M}\big(d(N + dB)\big)\big)$ binary operations; the precision loss is $O(2^m B)$ bits. Therefore the total precision loss when computing P is $O(dB)$ bits. The number of levels is $O(\log d)$, so the total cost is $O\big(\mathcal{M}\big(d(N + dB)\big) \log d\big)$ binary operations.

The computations are similar for the polynomial Q, with a different product tree. Each vertex at level $m + 1$ is a polynomial of the form $N_1 P_2 + N_2 P_1$ where P_i is a vertex of the product tree for P satisfying

$$\deg(P_i) \leq 2^m, \quad \log^+ |P_i| = O(2^m B),$$

and the polynomials N_i come from the m-th level, and satisfy

$$\deg(N_i) \leq 2^m - 1, \quad \log^+ |N_i| = O(C + 2^m B).$$

By induction, the m-th level can be computed to precision $N - O(C + 2^m B)$ within $O\big(\mathcal{M}\big(d(N + C + dB)\big)\big)$ binary operations. □

Lemma 2.4. *There exists an algorithm such that the following holds. Let $P \in \mathbb{Z}[X]$ be an irreducible polynomial of degree $d \geq 1$, let $(\alpha_i)_{1 \leq i \leq d}$ be the roots of P, and let $(t_i)_{1 \leq i \leq d}$ be complex numbers. Let $M, C \geq 1$ such that*

$$\log^+ |P| \leq M, \quad \text{and} \quad \log^+ |t_i| \leq C \quad \text{for every } i.$$

Let $N \geq 1$. Then, given P and approximations of the α_i, t_i, and $1/P'(\alpha_i)$ to precision N, the algorithm computes the polynomial Q of degree at most $d - 1$ interpolating the points (α_i, t_i) within $O\big(\mathcal{M}\big(d(N + C + dM + d \log d)\big) \log d\big)$ binary operations. The precision loss is $O(C + dM + d \log d)$ bits.

Proof. We write

$$Q = \sum_{i=1}^{d} \frac{t_i}{P'(\alpha_i)} \prod_{j \neq i} (X - \alpha_j).$$

We have $\log^+ |P'| \leq M + \log d$. The discriminant $\mathrm{Disc}(P)$ of P is the resultant of P and P', so we can write

$$UP + VP' = \mathrm{Disc}(P)$$

with $U, V \in \mathbb{Z}[X]$; the coefficients of U, V have expressions as determinants involving the coefficients of P and P', so by Hadamard's lemma, we have in particular

$$\log^+ |V| = O(dM + d \log d).$$

We have $\log^+ |\alpha_i| \leq M + \log(2)$ for every i, so

$$\log^+ \left| \frac{1}{P'(\alpha_i)} \right| = \log^+ \left| \frac{V(\alpha_i)}{\mathrm{Disc}(P)} \right| = O(dM + d \log d).$$

Therefore the precision loss when computing the complex numbers $z_i = t_i / P'(\alpha_i)$ is $O(C + dM + d \log d)$ bits; the total cost to compute the z_i is

$$O\big(d\mathcal{M}(N + C + dM + d \log d)\big)$$

binary operations. We conclude using Lemma 2.3. □

2.3 Recognizing integers in number fields

We conclude this section with estimates on the necessary precision to recognize integers in number fields.

We give two results according to the description of the number field. In the first description, the number field is $\mathbb{Q}(\alpha)$ where α is a root of some polynomial $P \in \mathbb{Z}[X]$ with bounded coefficients, and we want to recognize an element $x \in \mathbb{Z}[\alpha]$. This situation arises for instance when lifting from a finite field; not much is known about the number field itself. In the second description, we assume that an LLL-reduced basis of $\mathbb{Z}_F$ is known, and we want to recognize an element $x \in \mathbb{Z}_F$. The necessary precision is given in terms of the discriminant Δ_F of F and the absolute logarithmic height $h(x)$ of x.

Lemma 2.5. *There exist an algorithm and an absolute constant C such that the following holds. Let F be a number field of degree d defined by a monic irreducible polynomial $P \in \mathbb{Z}[X]$, and let $M \geq 1$ such that $\log^+ |P| \leq M$. Let α be a root of P in F. Let*

$$x = \sum_{j=0}^{d-1} \lambda_j \alpha^j \in \mathbb{Z}[\alpha]$$

with $\lambda_j \in \mathbb{Z}$ and $\log^+ |\lambda_j| \leq H$ for every j. Let $N \geq C(H + dM + d \log d)$. Then, given P and approximations of x, α and $1/P'(\alpha)$ to precision N in every complex embedding of F, the algorithm computes x within

$$O\big(\mathcal{M}\big(d(H + dM + d \log d) \log d\big)\big)$$

binary operations.

Proof. Denote the complex embeddings of F by $\sigma_1, \dots, \sigma_d$. Then the polynomial $Q = \sum_{j=0}^{d-1} \lambda_j X^j$ interpolates the points $(\sigma_i(\alpha), \sigma_i(x))$ for every $1 \leq i \leq d$. By assumption, we have for each i

$$\log^+ |\sigma_i(x)| \leq H + O(dM).$$

We are in the situation of Lemma 2.4: we can compute an approximation of Q with a precision loss of $O(H + dM + d \log d)$ bits. Therefore, for some choice of the constant C that we do not make explicit, the resulting precision is sufficient to obtain Q exactly by rounding the result to the nearest integers. □

Lemma 2.6. *There exist an algorithm and an absolute constant C such that the following holds. Let F be a number field of degree d and discriminant Δ_F. Let $(a_1, \dots, a_d)$ be an LLL-reduced basis of $\mathbb{Z}_F$, let $\sigma_1, \dots, \sigma_d$ be the complex embeddings of F, and let m_F be the matrix $(\sigma_i(a_j))_{1 \leq i,j \leq d}$. Let $x \in \mathbb{Z}_F$, and let $H \geq 1$ such that $h(x) \leq H$. Let $N \geq C(\log \Delta_F + dH + d \log d)$. Then, given approximations of $(\sigma_i(x))_{1 \leq i \leq d}$ and m_F^{-1} to precision N, the algorithm computes x within $O\big(d^2 \mathcal{M}(H + \log \Delta_F + d \log d)\big)$ binary operations.*

Proof. Since the basis $(a_1, \dots, a_d)$ is LLL-reduced, we have

$$\log^+ |m_F| = \log \Delta_F + O(d).$$

Let $\lambda_j \in \mathbb{Z}$ such that $x = \sum \lambda_j a_j$. Then, by definition of m_F, we have

$$\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_d \end{pmatrix} = m_F^{-1} \begin{pmatrix} \sigma_1(x) \\ \vdots \\ \sigma_d(x) \end{pmatrix}.$$

The determinant of m_F is Δ_F, so $|\det m_F| \geq 1$. By Hadamard's lemma, we have

$$\log^+ \left|m_F^{-1}\right| = \log \Delta_F + O(d \log d).$$

Since $h(x) \leq H$, we have $\sum_{i=1}^{d} \log^+ |\sigma_i(x)| \leq dH$. Therefore, for some choice of the constant C that we do not make explicit, we can recover the coefficients $\lambda_j \in \mathbb{Z}$ exactly. On average, we have $\log^+ |\sigma_i(x)| \leq H$, so the cost of each multiplication is on average $O\big(\mathcal{M}(H + M + d \log d)\big)$ binary operations. Therefore the total cost of the matrix-vector product is only $O\big(d^2 \mathcal{M}(H + M + d \log d)\big)$ binary operations. □

3 Siegel and Hilbert modular polynomials

In this section, we recall the definition of Siegel and Hilbert modular polynomials in genus 2. When studying their denominators, the structure of the corresponding rings of modular forms over $\mathbb{Z}$ plays a crucial role.

3.1 Siegel modular polynomials

Invariants on the Siegel moduli space. Let $\mathcal{H}_2$ be the set of symmetric 2×2 complex matrices τ such that $\operatorname{Im}\tau$ is positive definite. Our notation for the action of the symplectic group $\mathrm{Sp}_4(\mathbb{Z})$ has a natural action on $\mathcal{H}_2$ is as follows: for every $\gamma \in \mathrm{Sp}_4(\mathbb{Z})$ and $\tau \in \mathcal{H}_2$, write

$$\gamma\tau = (a\tau + b)(c\tau + d)^{-1} \quad \text{and} \quad \gamma^*\tau = c\tau + d, \quad \text{where} \quad \gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

We also write $\mathrm{Sp}_4(\mathbb{Z}) = \Gamma(1)$. For a subring $R \subset \mathbb{C}$, we denote by $\mathrm{MF}(\Gamma(1), R)$ the graded ring of Siegel modular forms of even weight defined over R.

The ring $\mathrm{MF}(\Gamma(1), \mathbb{C})$ is free over four generators h_4, h_6, h_{10}, h_{12}, where the subscript denotes the weight. We refer to [20, §7.1] for their explicit expressions in terms of theta constants. Following [20, §2.1], we define the Igusa invariants as follows:

$$j_1 = \frac{h_4 h_6}{h_{10}}, \quad j_2 = \frac{h_4^2 h_{12}}{h_{10}^2}, \quad j_3 = \frac{h_4^5}{h_{10}^2}.$$

The ring $\mathbb{Z}[h_4, h_6, h_{10}, h_{12}]$ is strictly contained in $\mathrm{MF}(\Gamma(1), \mathbb{Z})$, but is still reasonably large.

Lemma 3.1. *Let $f \in \mathrm{MF}(\Gamma(1), \mathbb{Z})$ of weight k. Then $12^k f \in \mathbb{Z}[h_4, h_6, h_{10}, h_{12}]$.*

Proof. Igusa [7] worked out a set of fourteen generators for the ring $\mathrm{MF}(\Gamma(1), \mathbb{Z})$. The lowest weight ones are

$$X_4 = 2^{-2} h_4, \quad X_6 = 2^{-2} h_6, \quad X_{10} = -2^{-12} h_{10}, \quad X_{12} = 2^{-15} h_{12}.$$

The result holds for these four generators. Straightforward computations using the formulas from [7, p. 153] prove that it also holds for the ten others. □

Siegel modular polynomials. Let $\ell \in \mathbb{Z}$ be a prime. We define the subgroup $\Gamma^0(\ell)$ of $\Gamma(1) = \mathrm{Sp}_4(\mathbb{Z})$ by

$$\Gamma^0(\ell) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma(1) \mid b = 0 \bmod \ell \right\}.$$

A set C_ℓ of representatives for the quotient $\Gamma^0(\ell)\backslash\Gamma(1)$ consists of the $\ell^3+\ell^2+\ell+1$ following matrices:

$$T_1(a,b,c) = \begin{pmatrix} -I_2 & \begin{matrix} a & b \\ b & c \end{matrix} \\ 0 & -I_2 \end{pmatrix} \quad \text{for } a,b,c \in [\![0,\ell-1]\!],$$

$$T_2(a,b,c) = \begin{pmatrix} 0 & -I_2 \\ I_2 & \begin{matrix} -a & -b \\ -b & -c \end{matrix} \end{pmatrix} \quad \text{for } a,b,c \in [\![0,\ell-1]\!] \text{ such that } ac = b^2 \bmod \ell,$$

$$T_3(a) = \begin{pmatrix} -1 & -a & 0 & 0 \\ 0 & 0 & -a & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} \quad \text{for } a \in [\![0,\ell-1]\!],$$

$$T_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 0 & 0 \end{pmatrix}.$$

The *Siegel modular polynomials* of level ℓ are the three polynomials

$$\Psi_{\ell,k} \in \mathbb{Q}(j_1,j_2,j_3)[X], \quad 1 \le k \le 3$$

such that for every $\tau \in \mathcal{H}_2$ where everything is well defined, we have

$$\begin{aligned} \Psi_{\ell,1}(j_1(\tau),j_2(\tau),j_3(\tau)) &= \prod_{\eta\in C_\ell} \big(X - j_1(\tfrac{1}{\ell}\eta\tau)\big), \\ \Psi_{\ell,2}(j_1(\tau),j_2(\tau),j_3(\tau)) &= \sum_{\eta\in C_\ell} j_2(\tfrac{1}{\ell}\eta\tau) \prod_{\eta'\in C_\ell\backslash\{\eta\}} \big(X - j_1(\tfrac{1}{\ell}\eta'\tau)\big), \\ \Psi_{\ell,3}(j_1(\tau),j_2(\tau),j_3(\tau)) &= \sum_{\eta\in C_\ell} j_3(\tfrac{1}{\ell}\eta\tau) \prod_{\eta'\in C_\ell\backslash\{\eta\}} \big(X - j_1(\tfrac{1}{\ell}\eta'\tau)\big). \end{aligned} \tag{1}$$

The degrees of the polynomials $\Psi_{\ell,k}$ in X are at most $\ell^3+\ell^2+\ell+1$, and their total degrees in j_1,j_2,j_3 are at most $10(\ell^3+\ell^2+\ell+1)/3$ by [9, Prop. 4.10]. The height of their coefficients is $O(\ell^3\log\ell)$ by [9, Thm. 1.1].

The denominator of Siegel modular polynomials. We call a polynomial $D_\ell \in \mathbb{Z}[j_1,j_2,j_3]$ a *denominator* of the Siegel modular polynomials $\Psi_{\ell,k}$ if for each $1 \le k \le 3$, we have

$$D_\ell \Psi_{\ell,k} \in \mathbb{Z}[j_1,j_2,j_3,X].$$

Our goal is to construct a denominator for the Siegel modular polynomials, given by an analytic formula. For every $\tau \in \mathcal{H}_2$, we define

$$g_\ell(\tau) = \prod_{\eta\in C_\ell} \det(\eta^*\tau)^{-20} h_{10}^2(\tfrac{1}{\ell}\eta\tau).$$

One can check that the function g_ℓ is independent of the choice of representatives of $\Gamma^0(\ell)\backslash\Gamma(1)$, and is a Siegel modular form of weight

$$w_\ell = 20(\ell^3 + \ell^2 + \ell + 1).$$

For every $\tau \in \mathcal{H}_2$ and $0 \le i \le \ell^3 + \ell^2 + \ell + 1$, we define $f_{\ell,k}^{(i)}(\tau)$ as the coefficient of X^i in the polynomial $g_\ell(\tau)\Psi_{\ell,k}(j_1(\tau), j_2(\tau), j_3(\tau))$. These functions are holomorphic on $\mathcal{H}_2$, as is easily seen from the formulæ (1) defining the modular polynomials, and also are Siegel modular forms of weight w_ℓ.

Proposition 3.2. *The modular forms g_ℓ and $f_{\ell,k}^{(i)}$ are defined over $\mathbb{Z}$.*

Proof. The modular form g_ℓ has an algebraic interpretation as follows. Start from a principally polarized abelian surface A and a basis ω of the space of differential forms $\Omega^1(A)$ on A, and let A_i be the abelian surfaces ℓ-isogenous to A. Let $f_i \colon A \to A_i$ be an ℓ-isogeny, and let ω_i be the basis of $\Omega^1(A_i)$ such that $f_i^*\omega_i = \omega$. Then

$$g_\ell(A, \omega) = \prod h_{10}^2(A_i, \omega_i).$$

Since the Hecke correspondence is defined over $\mathbb{Q}$, and the modular form h_{10} is also defined over $\mathbb{Q}$, the modular form g_ℓ is also defined over $\mathbb{Q}$. This is also the case of $f_{\ell,k}^{(i)}$, because the coefficients of Siegel modular equations are defined over $\mathbb{Q}$ as modular functions. Therefore we only have to show that their Fourier coefficients are algebraic integers.

Let f be a Siegel modular form of weight k, defined over $\mathbb{Z}$, and let $S \subset C_\ell$. Then we claim that the function

$$h(\tau) = \prod_{\eta \in S} \det(\eta^*\tau)^{-k} f(\tfrac{1}{\ell}\eta\tau)$$

has a Fourier expansion in terms of

$$\exp(2\pi i \tau_j/\ell^2), \quad 1 \le j \le 3, \quad \text{where } \tau = \begin{pmatrix} \tau_1 & \tau_3 \\ \tau_3 & \tau_2 \end{pmatrix}.$$

with coefficients in $\mathbb{Z}[\exp(2\pi i/\ell^2)]$. This implies Proposition 3.2 because g_ℓ and the $f_{\ell,k}^{(i)}$ are sums of such functions.

In order to compute the Fourier expansion of $h(\tau)$, we compute, for each $\eta \in S$, a matrix $\eta^R \in \Gamma(1)$ such that the transformation $\tau \mapsto \eta^R(\frac{1}{\ell}\eta\tau)$ leaves the cusp at infinity "invariant". More precisely, we require that

$$\eta^R \begin{pmatrix} a & b \\ \ell c & \ell d \end{pmatrix} = \begin{pmatrix} A_\eta & B_\eta \\ 0 & D_\eta \end{pmatrix}, \quad \text{where } \eta = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

To compute η^R, we proceed as follows. Let $u_1, u_2 \in \mathbb{Z}^4$ be the two columns of the 4×2 matrix $\binom{-\ell c}{a}$. Then $\langle u_1, u_2 \rangle = 0$, so u_1, u_2 are contained in an isotropic subspace $V \subset \mathbb{Q}^4$ of dimension 2. The two last lines of η^R are given by a basis of $\mathbb{Z}^4 \cap V$, and we complete them into a symplectic basis of $\mathbb{Z}^4$ to obtain η^R.

- If $\eta = T_1(a,b,c)$, we take

$$\eta^R = I_4, \quad D_\eta = -\ell I_2, \quad \det D_\eta = \ell^2.$$

- If $\eta = T_2(a,b,c)$, we take

$$\eta^R = J, \quad D_\eta = I_2, \quad \det D_\eta = 1.$$

- If $\eta = T_3(a)$, we take

$$\eta^R = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad D_\eta = \begin{pmatrix} -a & 1 \\ -\ell & 0 \end{pmatrix}, \quad \det D_\eta = \ell.$$

- If $\eta = T_4$, we take

$$\eta^R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \ell & 0 & 1 \\ -\ell & \ell & 1 & 1 \\ 1 & -1 & 0 & 0 \end{pmatrix}, \quad D_\eta = \begin{pmatrix} 2\ell & \ell \\ -1 & 0 \end{pmatrix}, \quad \det D_\eta = \ell.$$

Then we compute that

$$\begin{aligned} h(\tau) &= \prod_{\eta \in S} \det(\eta^* \tau)^{-k} \det\big(\eta^{R*}(\tfrac{1}{\ell}\eta\tau)\big)^{-k} f\big((A_\eta \tau + B_\eta) D_\eta^{-1}\big) \\ &= \prod_{\eta \in S} \ell^{2k} \det(D_\eta)^{-k} f\big((A_\eta \tau + B_\eta) D_\eta^{-1}\big). \end{aligned}$$

We develop $f((A_\eta\tau + B_\eta)D_\eta^{-1})$ using the Fourier expansion of f, and obtain a Fourier expansion with coefficients in $\mathbb{Z}[\exp(2\pi i/\ell^2)]$ in terms of $\exp(2\pi i \tau_j/\ell^2)$. Moreover each of the $\det(D_\eta)^{-k}\ell^{2k}$ is an integer. This proves the claim. □

The computations in the proof show that g_ℓ is divisible by $\ell^{20(2\ell^2+\ell+1)}$, but we do not need this fact. We finally define

$$D_\ell(\tau) = 12^{w_\ell} \frac{h_4(\tau)^{\lfloor w_\ell/6 \rfloor}}{h_{10}(\tau)^a h_4(\tau)^b} \prod_{\eta \in C_\ell} \det(\eta^* \tau)^{-20} h_{10}^2(\tfrac{1}{\ell}\eta\tau), \tag{2}$$

where a, b are such that $4\lfloor w_\ell/6 \rfloor + w_\ell = 10a + 4b$ with $0 \le b \le 4$. Let us show that D_ℓ is indeed a denominator of the Siegel modular polynomials of level ℓ.

Proposition 3.3. *$D_\ell \in \mathbb{Z}[j_1, j_2, j_3]$, and $D_\ell \Psi_{\ell,k} \in \mathbb{Z}[j_1, j_2, j_3, X]$ for $1 \le k \le 3$.*

Proof. By Proposition 3.2 and Lemma 3.1, we know that

$$12^{w_\ell} g_\ell \in \mathbb{Z}[h_4, h_6, h_{10}, h_{12}], \quad \text{and} \quad 12^{w_\ell} f_{\ell,k}^{(i)} \in \mathbb{Z}[h_4, h_6, h_{10}, h_{12}] \text{ for all } k, i.$$

Moreover, the equalities

$$h_4 h_6 = j_1 h_{10}, \quad h_4^2 h_{12} = j_2 h_{10}^2, \quad h_4^5 = j_3 h_{10}^2$$

show that for every modular form $f \in \mathbb{Z}[h_4, h_6, h_{10}, h_{12}]$ of weight k, we have

$$\frac{h_4^{\lfloor k/6 \rfloor} f}{h_{10}^a h_4^b} \in \mathbb{Z}[j_1, j_2, j_3]$$

where a, b are such that $4\lfloor k/6 \rfloor + k = 10a + 4b$ and $0 \leq b \leq 4$. See also the proof of [9, Lem. 4.7]. □

3.2 Hilbert modular polynomials

Invariants on Hilbert moduli spaces. Fix a real quadratic field K of discriminant Δ_K, and let $\mathbb{Z}_K$ be its ring of integers. We also fix a real embedding of K, and denote the other embedding by $x \mapsto \overline{x}$. Let $\mathbb{Z}_K^\vee = (1/\sqrt{\Delta})\,\mathbb{Z}_K$ be the dual of $\mathbb{Z}_K$ with respect to the trace form, and define the group

$$\Gamma_K(1) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}_2(K) \mid a, d \in \mathbb{Z}_K,\ b \in (\mathbb{Z}_K^\vee)^{-1},\ c \in \mathbb{Z}_K^\vee \right\}.$$

Write $\mathcal{H}_1 = \{z \in \mathbb{C} \mid \mathrm{Im}\, z > 0\}$. Our notation for the action of $\Gamma_K(1)$ on $\mathcal{H}_1^2$ is the following: for every $\gamma \in \Gamma_K(1)$, every $\alpha \in K$ and every $t = (t_1, t_2) \in \mathcal{H}_1^2$, we write

$$\gamma t = \left(\frac{a t_1 + b}{c t_1 + d}, \frac{\overline{a} t_2 + \overline{b}}{\overline{c} t_2 + \overline{d}} \right), \quad \alpha t = (\alpha t_1, \overline{\alpha} t_2), \quad \text{and} \quad \gamma^* \tau = (c t_1 + d)(\overline{c} t_2 + \overline{d}).$$

Each point of the quotient $\Gamma_K(1) \backslash \mathcal{H}_1^2$ corresponds to a principally polarized abelian surface over $\mathbb{C}$ with real multiplication by $\mathbb{Z}_K$. The involution of $\Gamma_K(1) \backslash \mathcal{H}_1^2$ given by

$$\sigma \colon (t_1, t_2) \mapsto (t_2, t_1)$$

exchanges the real multiplication embedding with its Galois conjugate.

One way to define generic invariants on Hilbert moduli spaces consists in pulling back invariants from the Siegel modular space by the forgetful map. Let (e_1, e_2) be a $\mathbb{Z}$-basis of $\mathbb{Z}_K$, and write

$$R_K = \begin{pmatrix} e_1 & e_2 \\ \overline{e}_1 & \overline{e}_2 \end{pmatrix} \in \mathrm{GL}_2(\mathbb{R}).$$

Then the *Hilbert embedding* is the map

$$\begin{array}{rccc} \Phi_K \colon & \mathcal{H}_1^2 & \to & \mathcal{H}_2 \\ & (t_1, t_2) & \mapsto & R^t \begin{pmatrix} t_1 & 0 \\ 0 & t_2 \end{pmatrix} R. \end{array} \tag{3}$$

One can check that Φ_K induces a map $\overline{\Phi}_K : \Gamma_K(1)\backslash\mathcal{H}_1^2 \to \Gamma(1)\backslash\mathcal{H}_2$, and that $\overline{\Phi}_K$ is independent of the choice of basis. Generically, the map $\overline{\Phi}_K$ is 2-1: for every $t \in \mathcal{H}_1^2$, we have $\overline{\Phi}_K(t) = \overline{\Phi}_K(\sigma(t))$. The image of $\overline{\Phi}_K$ inside $\Gamma(1)\backslash\mathcal{H}_2$ is algebraic, and is described in terms of Igusa invariants by the *Humbert equation*

$$H_K(j_1, j_2, j_3) = 0, \quad H_K \in \mathbb{Z}[j_1, j_2, j_3].$$

The pullback of Igusa invariants by $\overline{\Phi}_K$ are symmetric invariants, and could be used to define Hilbert modular polynomials. However, we do not know in general how to relate j_1, j_2, j_3 to a set of generators for the ring $\mathrm{MF}(\Gamma_K(1), \mathbb{Z})$ of symmetric Hilbert modular forms of even weight, so we are unable to give a general formula for the denominator of Hilbert modular polynomials in Igusa invariants.

When the structure of $\mathrm{MF}(\Gamma_K(1), \mathbb{Z})$ is known, we could in principle derive such a formula, but it is better to design other invariants in terms of the generating set. For instance, when $K = \mathbb{Q}(\sqrt{5})$ (resp. $K = \mathbb{Q}(\sqrt{2})$), the ring $\mathrm{MF}(\Gamma_K(1), \mathbb{Z})$ is generated by four elements G_2, F_6, F_{10}, F_{12} (resp. three elements G_2, F_4, F_6), where the subscripts denote the weights [17]. In these cases we define the Gundlach invariants g_1, g_2 as

$$g_1 = \frac{G_2^5}{F_{10}},\ g_2 = \frac{G_2^2 F_6}{F_{10}} \quad \Big(\text{resp. } g_1 = \frac{G_2^2}{F_4},\ g_2 = \frac{G_2 F_6}{F_4^2}\Big).$$

For simplicity, we concentrate on Hilbert modular polynomials in Gundlach invariants for $K = \mathbb{Q}(\sqrt{5})$ in the sequel, and highlight only the differences that would appear in the case of Igusa invariants. Then, the analogue of Lemma 3.1 is as follows.

Lemma 3.4. *Let $K = \mathbb{Q}(\sqrt{5})$. Then for every $f \in \mathrm{MF}(\Gamma_K(1), \mathbb{Z})$ of weight k, we have $2^k f \in \mathbb{Z}[G_2, G_6, F_{10}]$.*

Proof. The ring $\mathrm{MF}(\Gamma_K(1), \mathbb{Z})$ is generated by G_2, F_6, F_{10}, and

$$F_{12} = \frac{1}{4}(F_6^2 - G_2 F_{10}). \qquad \square$$

Hilbert modular polynomials. Let ℓ be a prime that splits in $K = \mathbb{Q}(\sqrt{5})$ in two principal ideals generated by totally positive elements $\beta, \overline{\beta} \in \mathbb{Z}_K$. Define the subgroup $\Gamma_K^0(\beta)$ of $\Gamma_K(1)$ by

$$\Gamma_K^0(\beta) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_K(1) \mid b = 0 \bmod \beta \right\}.$$

A set C_β of representatives for the quotient $\Gamma_K^0(\beta)\backslash\Gamma(1)$ consists of the $\ell + 1$ following matrices:

$$\begin{pmatrix} 0 & \sqrt{\Delta_K} \\ -1/\sqrt{\Delta_K} & 0 \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} 1 & a\sqrt{\Delta_K} \\ 0 & 1 \end{pmatrix} \quad \text{for } a \in [\![0, \ell - 1]\!].$$

A set of representatives for the quotient $\Gamma_K^0(\beta)\backslash(\Gamma(1)\rtimes\langle\sigma\rangle)$ is $C_\beta^\sigma = C_\beta \cup C_\beta\sigma$. The Hilbert modular polynomials of level β in Gundlach invariants are the two polynomials

$$\Psi_{\beta,k} \in \mathbb{Q}(g_1, g_2)[X], \quad 1 \leq k \leq 2$$

such that for every $t \in \mathcal{H}_1^2$, we have

$$\Psi_{\beta,1}(g_1(t), g_2(t)) = \prod_{\eta \in C_\beta^\sigma} \big(X - g_1(\tfrac{1}{\beta}\eta t)\big),$$

$$\Psi_{\beta,2}(g_1(t), g_2(t)) = \sum_{\eta \in C_\beta^\sigma} g_2(\tfrac{1}{\beta}\eta t) \prod_{\eta' \in C_\beta^\sigma \setminus \{\eta\}} \big(X - g_1(\tfrac{1}{\beta}\eta' t)\big).$$

The degrees of the polynomials $\Psi_{\beta,k}$ in X are at most $2\ell+2$, and their total degrees in g_1, g_2 are at most $10(\ell+1)/3$ by [9, Prop. 4.11]. The height of their coefficients is $O(\ell \log \ell)$ by [9, Thm. 1.1].

The denominator of Hilbert modular polynomials. We call $D_\beta \in \mathbb{Z}[g_1, g_2]$ a *denominator* of the Hilbert modular polynomials $\Psi_{\beta,k}$ if for each $1 \leq k \leq 2$, we have

$$D_\beta \Psi_{\beta,k} \in \mathbb{Z}[g_1, g_2].$$

We now construct such a denominator in the case $K = \mathbb{Q}(\sqrt{5})$; the procedure is easily copied in the case $K = \mathbb{Q}(\sqrt{2})$. For every $t \in \mathcal{H}_1^2$, we define

$$g_\beta(t) = \prod_{\eta \in C_\beta^\sigma} (\eta^* t)^{-20} h_{10}^2(\tfrac{1}{\beta}\eta t).$$

We can show that g_β is independent of the choice of representatives of the quotient $\Gamma_K^0(\beta)\backslash\Gamma_K(1)$, and that g_β is a symmetric Hilbert modular form of weight

$$w_\beta = 20(2\ell+2).$$

As above, for every $0 \leq i \leq 2\ell+2$ and $1 \leq k \leq 2$, we define $f_{\beta,k}^{(i)}(t)$ as the coefficient of X^i in the polynomial $g_\beta(t)\Psi_{\beta,k}(g_1(t), g_2(t))$. It is also a symmetric modular form of weight w_β.

Proposition 3.5. *The modular forms g_β and $f_{\beta,k}^{(i)}$ are defined over $\mathbb{Z}$.*

Proof. As before, both the Hecke correspondence and h_{10} are defined over $\mathbb{Q}$, so g_β is defined over $\mathbb{Q}$. This is also the case of $f_{\beta,k}^{(i)}$ because the coefficients of Hilbert modular equations are defined over $\mathbb{Q}$ as modular functions.

As in Proposition 3.2, it is enough to show the following: if f is a Hilbert modular form of weight k and S is a subset of C_β^σ, then the function

$$h(t) = \prod_{\eta \in S} (\eta^* t)^{-k} f(\tfrac{1}{\beta}\eta t)$$

has a Fourier expansion in terms of the $\exp(2\pi i(nt_1 + \overline{n}t_2)/\ell)$ for $n \in \mathbb{Z}_K$ such that $n \gg 0$, with coefficients in $\mathbb{Z}[\exp(2\pi i/\ell)]$. The computations are easier than in the Siegel case due to the simpler form of coset representatives:

- If $\eta = \begin{pmatrix} 0 & \sqrt{\Delta_K} \\ -1/\sqrt{\Delta_K} & 0 \end{pmatrix}$, then we make η act again, and compute that

$$(\eta^* t)^{-k} f(\tfrac{1}{\beta}\eta t) = \ell f(\beta t).$$

- If $\eta = \begin{pmatrix} 1 & a\sqrt{\Delta_K} \\ 0 & 1 \end{pmatrix}$, then we directly have

$$(\eta^* t)^{-k} f(\tfrac{1}{\beta}\eta t) = f(\tfrac{1}{\beta}(t + (a\sqrt{\Delta_K}, -a\sqrt{\Delta_K})).$$

Therefore the Fourier expansion of h must have integer coefficients. □

For every $t \in \mathcal{H}_1^2$, we define

$$D_\beta(t) = 2^{w_\beta} \frac{G_2(t)^{2\lfloor w_\beta/6 \rfloor}}{F_{10}(t)^a G_2(t)^b} \prod_{\eta \in C_\beta^\sigma} (\eta^* t)^{-20} h_{10}^2(\tfrac{1}{\beta}\eta\tau). \tag{4}$$

where a, b are such that $2\lfloor w_\beta/6 \rfloor + w_\beta = 10a + 2b$ with $0 \le b \le 4$.

Proposition 3.6. *Let $K = \mathbb{Q}(\sqrt{5})$ and β as above. Then D_β is a denominator of the Hilbert modular polynomials of level β.*

Proof. By Proposition 3.5 and Lemma 3.4, we know that

$$2^{w_\beta} g_\beta \in \mathbb{Z}[G_2, F_6, F_{10}], \quad \text{and} \quad 2^{w_\eta} f_{\beta,k}^{(i)} \in \mathbb{Z}[G_2, F_6, F_{10}] \text{ for all } k, i.$$

Moreover, the equalities

$$G_2^2 F_6 = g_2 F_{10}, \quad G_2^5 = g_1 F_{10}$$

show that for every modular form $f \in \mathbb{Z}[G_2, F_6, F_{10}]$ of weight k, we have

$$\frac{G_2^{2\lfloor k/6 \rfloor} f}{F_{10}^a G_2^b} \in \mathbb{Z}[g_1, g_2]$$

where a, b are such that $4\lfloor k/6 \rfloor + k = 10a + 2b$ and $0 \le b \le 4$. □

In the general case of Hilbert modular equations in Igusa invariants where we are not able do determine a denominator a priori, we will have to recognize general elements in number fields, instead of integers, from complex approximations. Although the height of these rational numbers is $O_K(\ell \log \ell)$, the dependence of the implied constant on K is unknown. A possible strategy is to double the precision until two consecutive rational reconstructions give the same result: this is guaranteed to terminate within the same runtime bound, up to constants. However, correctness is subject to the heuristic that the algorithm will not stop with an incorrect result before reaching the correct precision.

4 Computing theta functions

In this section, we study the critical part of the evaluation algorithm in terms of practical runtime, namely the computation of theta functions at high precision at various points in $\mathcal{H}_2$. Using an algorithm of Dupont [4, §10.2], this can be heuristically done in quasi-linear time for a *fixed* τ satisfying the following conditions:

$$\begin{aligned} |x_j| &\leq \frac{1}{2} \quad \text{for each } 1 \leq j \leq 3, \\ 2\,|y_3| &\leq y_1 \leq y_2, \\ |z_j| &\geq 1 \quad \text{for } j \in \{1, 2\}, \end{aligned} \tag{5}$$

where we denote

$$\tau = \begin{pmatrix} z_1 & z_3 \\ z_3 & z_2 \end{pmatrix}, \quad x_j = \operatorname{Re} z_j, \quad y_j = \operatorname{Im} z_j.$$

Note that [4, Conjecture 9.1] holds under these conditions by [10]: theta constants at τ can be computed using Borchardt means with good choices of square roots.

Dupont's algorithm is based on Newton iterations, but their convergence has not yet been proved. Here, we make the fundamental assumption that these Newton iterations do converge, and even converge *uniformly* on a compact set. This must be the case if the Jacobian of the Newton system is invertible and the functions involved are analytic.

Hypothesis 4.1. *There exists an algorithm such that the following holds. Let $\tau \in \mathcal{H}_2$ and $N \geq 1$. Assume that τ satisfies conditions (5) together with the bound*

$$\max\{y_1, y_2\} \leq 10.$$

Then, given an approximation of τ to precision N, the algorithm computes squares of theta constants at τ in $O(\mathcal{M}(N) \log N)$ binary operations with a precision loss of $O(1)$ bits.

The subset of $\mathcal{H}_2$ defined by the conditions in hypothesis 4.1 is compact. Derivatives of theta constants are uniformly bounded on this compact set, hence the precision loss of $O(1)$ bits.

Using this assumption, we first describe an algorithm to compute theta functions in *uniform* quasi-linear time at a given τ that belongs to the fundamental domain $\mathcal{F}_2$, using techniques similar to the genus 1 case [5, Thm. 5]. Second, for any $\tau \in \mathcal{H}_2$, we adapt the classical reduction algorithm to find $\tau' \in \Gamma(1)\tau$ very close to $\mathcal{F}_2$. Put together, this allows us to evaluate theta functions at every $\tau \in \mathcal{H}_2$, and to bound the precision losses involved.

4.1 Computing theta constants on $\mathcal{F}_2$

There are two easy cases:

1. If τ belongs to the compact set from hypothesis 4.1, then we use Dupont's algorithm directly.
2. If y_1 and y_2 satisfy $\min\{y_1, y_2\} \geq CN$, where C is an absolute constant, then we use the naive algorithm to compute theta constants at τ within $O(\mathcal{M}(N))$ binary operations.

For other values of $\tau \in \mathcal{F}_2$, we fall back to one of these two cases using duplication formulas. For every $\tau \in \mathcal{H}_2$, write

$$D_1(\tau) = \frac{\tau}{2}, \qquad D_2(\tau) = \begin{pmatrix} 2z_1(\tau) & z_3(\tau) \\ z_3(\tau) & \frac{1}{2}z_2(\tau) \end{pmatrix}.$$

Lemma 4.2. *Let $\tau \in \mathcal{H}_2$ satisfying conditions (5).*

1. *If $D_1(\tau)$ satisfies conditions (5), then $(\theta_j^2(\tau))_{j\in\{0,1,2,3\}}$ is obtained from $(\theta_j^2(D_1(\tau)))_{j\in\{0,1,2,3\}}$ by a Borchardt iteration with good choice of roots.*
2. *If $D_2(\tau)$ satisfies conditions (5), except that the real part of z_1 is allowed to be smaller than 1 instead of 1/2, then $(\theta_j^2(\tau))_{j\in\{0,2,4,6\}}$ is obtained from $(\theta_j^2(D_2(\tau)))_{j\in\{0,2,4,6\}}$ by a Borchardt iteration with good choice of roots.*

Proof. The first item is the classical duplication formula: the choice of roots is given by the theta constants $\theta_j(D_1(\tau))$ for $0 \leq j \leq 3$, and they are in good position by [20, Prop. 7.7]. For the second item, apply the theta transformation formula [6, Thm. 2 p. 175 and Cor. p. 176] for the symplectic matrix

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The theta constants $\theta_j(D_2(\tau))$ for $j \in \{0, 2, 4, 6\}$ also are in good position by [10, Lem. 5.2]. □

Proposition 4.3. *There exists an algorithm which, given $\tau \in \mathcal{F}_2$ and $N \geq 1$, computes the squares of theta constants at τ to precision N within $O(\mathcal{M}(N)\log N)$ binary operations, uniformly in τ.*

Proof. First, we let k_2 be the smallest integer such that

$$2^{k_2} y_1(\tau) \geq \min\{CN,\, 2^{-k_2-2} y_2(\tau)\}$$

where C is an absolute constant. Let τ' be the matrix obtained after applying k_2 times D_2 to τ and reducing the real part at each step. In order to compute theta constants at τ to precision N, we will compute theta constants at τ' to some precision $N' \geq N$, then apply k_2 times the duplication formula from Lemma 4.2. We have $k_2 \in O(\log N)$, and the total precision loss taken in extracting square roots is $O(N)$, so the total precision loss is $O(N)$ bits: hence we can choose $N' = C'N$ where C' is an absolute constant.

Two cases arise now. If $y_1(\tau') \geq CN$, then we also have $y_2(\tau') \geq CN$, so we can compute theta constants at τ' to precision N' using $O(\mathcal{M}(N))$ operations with the naive algorithm. Otherwise, we have

$$y_1(\tau') \leq y_2(\tau') \leq 4y_1(\tau') \leq 4CN.$$

Therefore we can find an integer $k_1 \in O(\log N)$ such that $D_1^{k_1}(\tau')$ belongs to the compact set defined in hypothesis 4.1. The total precision loss arising from the duplication formula for D_1 is $O(1)$ bits per step, so we recover theta constants at τ' to precision N within $O(\mathcal{M}(N)\log N)$ binary operations. □

4.2 The approximate reduction algorithm

In order to evaluate theta constants at a given $\tau \in \mathcal{H}_2$, our strategy is to reduce τ to $\mathcal{F}_2$ and to compute theta constants there. However, the classical reduction algorithm is based on inequalities, and this causes problems on the boundary when the input is inexact. Therefore, we rather describe reduction algorithms to *neighborhoods* of $\mathcal{F}_2$; we still write inequalities, but they should be understood as *inclusions of intervals*. Then we show the validity of the reduction algorithm on inexact input provided that the precision remains high enough.

We start by defining neighborhoods of $\mathcal{F}_2$. Fix $\varepsilon > 0$, and let

$$Y = \begin{pmatrix} y_1 & y_3 \\ y_3 & y_2 \end{pmatrix}$$

be a symmetric 2×2 real matrix. Assume that Y is positive definite. We say that Y is *ε-Minkowski reduced* if

$$y_1 \leq (1+\varepsilon)y_2 \quad \text{and} \quad -\varepsilon y_1 \leq 2y_3 \leq (1+\varepsilon)y_1.$$

Let $\Sigma \subset \Gamma(1)$ be the set of 19 matrices defining the boundary of $\mathcal{F}_2$. We define the neighborhood $\mathcal{F}_2^\varepsilon$ of $\mathcal{F}_2$ as the set of all matrices $\tau \in \mathcal{H}_2$ such that

1. $\mathrm{Im}(\tau)$ is ε-Minkowski reduced.
2. $|\mathrm{Re}(\tau)| \leq 1/2 + \varepsilon$.
3. $|\det(\sigma^*\tau)| \geq 1 - \varepsilon$ for every $\sigma \in \Sigma$.

The fundamental domain $\mathcal{F}_2$ corresponds to the case $\varepsilon = 0$.

We now describe the approximate reduction algorithm. The input is $\tau \in \mathcal{H}_2$ to precision $N \geq 1$, and the output is $\tau' \in \mathcal{F}_2^\varepsilon$ together with $\gamma \in \Gamma(1)$ such that $\tau' = \gamma\tau$. We assume that the precision is greater than $|\log \varepsilon| + 1$ at any time. If we run out of precision, we stop and output "failure".

Algorithm 4.4 (Reduction to $\mathcal{F}_2^\varepsilon$)**.** Start with $\tau' = \tau$ and iterate the following three steps until $\tau' \in \mathcal{F}_2^\varepsilon$, keeping track of a matrix $\gamma \in \Gamma(1)$ such that $\tau' = \gamma\tau$:

1. Reduce $\mathrm{Im}(\tau')$ such that it becomes ε-Minkowski reduced.
2. Reduce $\mathrm{Re}(\tau')$ such that $|\mathrm{Re}(\tau')| \leq 1/2 + \varepsilon$.

3. Apply $\sigma \in \Sigma$ such that $|\det \sigma^*(\tau')|$ is at most $1-\varepsilon/2$ and minimal, if such a σ exists.
4. Update $\gamma \in \Gamma(1)$ and recompute $\tau' = \gamma\tau$.

In order to analyze Algorithm 4.4, we mimic Streng's analysis of the exact reduction algorithm [20, §II.5.3]. For $\tau \in \mathcal{H}_2$, we define

$$\Lambda(\tau) = \log \max\{2, |\tau|, \det(\operatorname{Im}\tau)^{-1}\}.$$

Denote by $\lambda_1(\tau) \leq \lambda_2(\tau)$ the two eigenvalues of $\operatorname{Im}(\tau)$, and by $m_1(\tau) \leq m_2(\tau)$ the successive minima of $\operatorname{Im}(\tau)$ on the lattice $\mathbb{Z}^2$. By [20, (5.4) p. 54], we always have

$$\frac{3}{4} m_1(\tau) m_2(\tau) \leq \det \operatorname{Im}(\tau) \leq m_1(\tau) m_2(\tau), \tag{6}$$

so that

$$\log \max\{\lambda_1(\tau)^{-1}, \lambda_2(\tau), m_1(\tau)^{-1}, m_2(\tau)\} \in O(\Lambda(\tau)).$$

First, we detail the Minkowski reduction step.

Lemma 4.5. *There exists an algorithm and an absolute constant C such that the following holds. Let $\tau \in \mathcal{H}_2$ and $\varepsilon > 0$. Then, given an approximation of τ to precision $N \geq C(\Lambda(\tau) + |\log \varepsilon|)$, the algorithm computes a matrix $U \in \mathrm{SL}_2(\mathbb{Z})$ such that $U^t \operatorname{Im}(\tau) U$ is ε-Minkowski reduced within $O(\mathcal{M}(N)\log N)$ binary operations.*

Proof. Write $\operatorname{Im}(\tau) = R^t R$, and consider the matrix R' obtained by rounding the coefficients of $2^N R$ to the nearest integers. If C is chosen appropriately, then the matrix R' is still invertible. We apply a quasi-linear version of the LLL algorithm to R' [18], and obtain a reduced basis of the lattice $R'\mathbb{Z}^2$ within $O(\mathcal{M}(N)\log N)$ binary operations . The base change matrix $U \in \mathrm{SL}_2(\mathbb{Z})$ must satisfy

$$\log |U| = O(\Lambda(\tau))$$

by [20, Lem. 5.6]. Therefore, the matrix $U^t \operatorname{Im}(\tau) U$ will be ε-Minkowski reduced if C is large enough. □

Then, we bound precision losses during Algorithm 4.4.

Lemma 4.6. *Let $\tau, \tau' \in \mathcal{H}_2$, and assume that there exists $\gamma \in \Gamma(1)$ such that $\tau' = \gamma\tau$. Then we have*

$$\log^+ \max\{|\gamma^*\tau|, |(\gamma^*\tau)^{-1}|\} = O(\max\{\Lambda(\tau), \Lambda(\tau')\}),$$
$$\log |\gamma| = O(\max\{\Lambda(\tau), \Lambda(\tau')\}).$$

Proof. Let R be a real matrix such that $R^t R = \operatorname{Im}(\tau)$. Then we have

$$\operatorname{Im}(\tau') = (\gamma^*\tau)^{-t} \operatorname{Im}(\tau) (\gamma^*\overline{\tau})^{-1} = R'^t \overline{R'}$$

with $R' = R(\gamma^*\tau)^{-1}$. Since $|R| \leq |\mathrm{Im}(\tau)|^{1/2}$ and $|R'| \leq |\mathrm{Im}(\tau')|^{1/2}$, we obtain

$$|\gamma^*\tau| = \left|R'^{-1}R\right| \leq 2\frac{|R'|}{\det(R')}\,|R|$$

so $\log^+|\gamma^*\tau| = O(\max\{\Lambda(\tau), \Lambda(\tau')\})$, and similarly for $(\gamma^*\tau)^{-1}$.

It remains to bound $|\gamma|$. If c, d denote the two lower blocks of γ, then we have $\mathrm{Im}(\gamma^*\tau) = c\,\mathrm{Im}(\tau)$. Therefore $\log^+|c| = O(\max\{\Lambda(\tau), \Lambda(\tau')\})$, and in turn $\log^+|d| \leq \log^+(|c\tau| + |\gamma^*\tau|) = O(\max\{\Lambda(\tau), \Lambda(\tau')\})$. Finally, we bound the upper blocks a and b of γ using the relation $a\tau + b = \tau'(c\tau + d)$. □

Lemma 4.7. *There is an absolute constant C such that the following holds. Let $\tau \in \mathcal{H}_2$ and $\varepsilon > 0$, and assume that the precision during Algorithm 4.4 remains greater than $|\log\varepsilon| + 1$. Then the number of iterations is $O(\Lambda(\tau))$. Moreover, during the algorithm, the quantities $|\log(|\det(\gamma^*\tau)|)|$, $\Lambda(\tau')$ and $\log|\gamma|$ remain in $O(\Lambda(\tau))$.*

Proof. The number of iterations is $O(\Lambda(\tau))$ by [20, Prop. 5.16]: observe that [20, Lem. 5.14 and 5.15] still apply, because $\det\mathrm{Im}(\tau')$ is strictly increasing in Algorithm 4.4. The proof of [20, Lem. 5.17] also applies to Algorithm 4.4 with slightly worse constants. This shows the bound $O(\Lambda(\tau))$ on $\log|\tau'|$ and $\log|\det(\gamma^*\tau)|$.

During the algorithm, we have $\log^+ m_2(\tau') = O(\Lambda(\tau))$ by [20, Lem. 5.12]. Moreover $\det\mathrm{Im}(\tau') \geq \det\mathrm{Im}(\tau)$, so

$$m_1(\tau')^{-1} \leq \frac{m_2(\tau')}{\det\mathrm{Im}(\tau')} \leq \frac{m_2(\tau')}{\det\mathrm{Im}(\tau)} \leq \frac{4m_2(\tau')}{3m_1(\tau)^2}$$

by (6). Therefore we also have $\Lambda(\tau') = O(\Lambda(\tau))$. The remaining bounds follow from Lemma 4.6. □

Proposition 4.8. *There is an absolute constant C such that the following holds. Let $\tau \in \mathcal{H}_2$ and $\varepsilon > 0$. Then, given an approximation of τ to precision $N \geq C(\Lambda(\tau) + |\log\varepsilon|)$ as input, Algorithm 4.4 does not run out of precision, and computes a matrix $\gamma \in \Gamma(1)$ such that $\gamma\tau \in \mathcal{F}_2^\varepsilon$ and $\log|\gamma| = O(\Lambda(\tau))$. It costs $O(\mathcal{M}(N)N\log N)$ binary operations.*

Proof. By Lemma 4.7, there is a constant C' such that $\log|\gamma| \leq C'\Lambda(\tau)$ during the execution of Algorithm 4.4 as long as the absolute precision is at least $|\log\varepsilon| + 1$. Therefore, if C is chosen appropriately, step 4 in Algorithm 4.4 ensures that the absolute precision is at least $|\log\varepsilon| + 1$ at every step. Hence the estimate on $\log|\gamma|$ and $\Lambda(\tau')$ remains valid until the end of the algorithm, and we can perform the approximate Minkowski reductions using Lemma 4.5.

By Lemma 4.7, there are $O(\Lambda(\tau))$ steps in Algorithm 4.4, and by Lemma 4.5, each step costs $O(\mathcal{M}(N)\log N)$ binary operations. Hence the cost is overall $O(\mathcal{M}(N)N\log N)$ binary operations. When the algorithm stops, the absolute precision is still greater than $|\log\varepsilon| + 1$, so the final τ' must belong to $\mathcal{F}_2^\varepsilon$. □

Given $\tau' \in \mathcal{F}_2^\varepsilon$, we can increase the imaginary parts of the coefficients slightly to obtain $\tau'' \in \mathcal{H}_2$ that satisfies conditions (5), and such that

$$|\tau'' - \tau'| \leq C\varepsilon\,|\tau'|$$

for some absolute constant C.

Corollary 4.9. *Under hypothesis 4.1, there exist an algorithm and an absolute constant C such that the following holds. Let $\tau \in \mathcal{H}_2$ and $N \geq 1$. Then, given an approximation of τ to precision $N + C\Lambda(\tau)$, the algorithm computes*

1. *a matrix $\gamma \in \mathrm{Sp}_4(\mathbb{Z})$ such that $\log|\gamma| = O(\Lambda(\tau))$,*
2. *a matrix $\tau'' \in \mathcal{F}_2$ such that τ'' is an approximation of $\gamma\tau$ to precision N,*
3. *an approximation of squares of theta constants at $\gamma\tau$ to precision N,*

within

$$O\big(\mathcal{M}(\Lambda(\tau))\Lambda(\tau)\log\Lambda(\tau) + \mathcal{M}(N)\log N\big)$$

binary operations.

Proof. Fix $\varepsilon = 0.01$, for instance. First, we apply Proposition 4.8 to compute γ such that $\gamma\tau \in \mathcal{F}_2^\varepsilon$, using $O\big(\mathcal{M}(\Lambda(\tau))\Lambda(\tau)\log\Lambda(\tau)\big)$ binary operations. Then, we recompute τ' to high precision, and reduce it further if necessary to land in $\mathcal{F}_2^{\varepsilon'}$ where $\varepsilon' = 2^{-N}\exp(-C\Lambda(\tau))$ for some appropriate constant C. This costs $O(\mathcal{M}(N+\Lambda(\tau)))$ binary operations. Finally we compute τ'' which satisfies conditions (5) and is close to τ'; the matrix τ'' is still an approximation of $\gamma\tau$, with $O(\Lambda(\tau))$ bits of precision lost. We output theta constants at τ'' to precision $N + O(1)$, which can be computed in time $O(\mathcal{M}(N)\log N)$ by Proposition 4.3. □

5 Computing period matrices

Let F be a number field of degree d, and fix a complex embedding σ of F. In this section, we investigate the first step of the algorithm to evaluate modular polynomials: given Igusa invariants $(j_1, j_2, j_3) \in F^3$, compute a period matrix $\tau \in \mathcal{F}_2$ with Igusa invariants $\sigma(j_1), \sigma(j_2), \sigma(j_3)$. We may assume that $j_3 \neq 0$: otherwise, modular polynomials are not defined at (j_1, j_2, j_3). Then τ can be computed in quasi-linear time using Borchardt means [4, §9.2.3]; our goal is to bound the precision loss in the process.

During the algorithm, we will consider finitely many algebraic complex numbers constructed from $\sigma(j_1), \sigma(j_2), \sigma(j_3)$. Let $B_\sigma \geq 0$ such that

$$|\log(|\theta|)| \leq B_\sigma$$

for each nonzero θ in this finite family of algebraic numbers. If H denotes the height of (j_1, j_2, j_3), then we can take $B_\sigma = O(dH)$; but in fact the sum of these bounds over all the complex embeddings of F is also $O(dH)$. A typical example of how we use B_σ is as follows.

Lemma 5.1. *There exists an algorithm such that the following holds. Let $j_1, j_2, j_3 \in F$ such that $j_3 \neq 0$, let σ be a complex embedding of F, and define B_σ as above. Let $N \geq 1$. Then, given approximations of $\sigma(j_k)$ for $1 \leq k \leq 3$ to precision N, the algorithm computes a genus 2 hyperelliptic curve $\mathcal{C}$ over $\mathbb{C}$ with Igusa invariants $\sigma(j_1), \sigma(j_2), \sigma(j_3)$ within $O(\mathcal{M}(N + B_\sigma))$ binary operations, with a precision loss of $O(B_\sigma)$ bits.*

Proof. Use Mestre's algorithm [13]. This algorithm involves $O(1)$ elementary operations with algebraic numbers constructed from $\sigma(j_k)$ for $1 \leq k \leq 3$, hence the estimates on the running time and the precision loss. □

We first prove that the period matrix $\tau \in \mathcal{F}_2$ of $\mathcal{C}$ is bounded in terms of B_σ. This is done by looking at theta quotients at τ. We use the traditional notation for genus 2 theta constants, used in [4, 20]: the even ones are

$$\theta_j(\tau) \quad \text{for } j \in \{0, 1, 2, 3, 4, 6, 8, 9, 12, 15\}.$$

Lemma 5.2. *Let $\mathcal{C}$ be as in Lemma 5.1, and let $\tau \in \mathcal{F}_2$ be its period matrix. Then we have*

$$|\tau| = O(B_\sigma).$$

Proof. The theta quotients $\theta_j(\tau)/\theta_0(\tau)$ are algebraic numbers constructed from the coefficients of $\mathcal{C}$, and are nonzero if $j \neq 15$. In particular, we have

$$\left| \log \left(\left| \frac{\theta_j(\tau)}{\theta_0(\tau)} \right| \right) \right| = O(B_\sigma) \quad \text{for } j \in \{4, 8\}.$$

Write $\tau = \begin{pmatrix} \tau_1 & \tau_3 \\ \tau_3 & \tau_2 \end{pmatrix}$. By [20, Prop. 7.7], we have

$$\begin{aligned} |\theta_0(\tau) - 1| &< 0.405, \\ |\theta_4(\tau)/\exp(i\pi\tau_1/4) - 1| &< 0.368, \\ |\theta_8(\tau)/\exp(i\pi\tau_2/4) - 1| &< 0.368. \end{aligned}$$

Therefore both $\mathrm{Im}(\tau_1)$ and $\mathrm{Im}(\tau_2)$ are in $O(B_\sigma)$, hence also $|\mathrm{Im}(\tau_3)|$ because $\det \mathrm{Im}(\tau) > 0$. Since $|\mathrm{Re}(\tau)| \leq 1/2$, the result follows. □

In order to compute τ from its theta quotients, we use Borchardt means [4, §9.2.3]. Define the matrices $J, M_1, M_2, M_3 \in \mathrm{Sp}_4(\mathbb{Z})$ whose action on $\tau \in \mathcal{H}_2$ is given by

$$J\tau = -\tau^{-1}, \quad M_1\tau = \tau + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad M_2\tau = \tau + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad M_3\tau = \tau + \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Let $\gamma_i = (JM_i)^2$ for $1 \leq i \leq 3$. Given theta quotients at τ, we compute the values of the modular functions

$$b_j = \theta_j^2/\theta_0^2, \quad 1 \leq j \leq 3$$

at τ and $\gamma_i\tau$ for $1 \leq i \leq 3$. We obtain the quantities

$$\frac{1}{\theta_0^2(\gamma_i\tau)}, \quad 1 \leq i \leq 3$$

as the limits of Borchardt sequences with *good sign choices* [10] starting from the tuples $(1, b_1(\gamma_i\tau), b_2(\gamma_i\tau), b_3(\gamma_i\tau))$. Finally, we use that

$$\theta_0^2(\gamma_1\tau) = -i\tau_1\theta_4^2(\tau), \quad \theta_0^2(\gamma_2\tau) = -i\tau_2\theta_8^2(\tau), \quad \theta_0^2(\gamma_3\tau) = -\det(\tau)\theta_0^2(\tau). \quad (7)$$

In order to bound the complexity of this algorithm, we need estimates on the convergence of the Borchardt sequences above that are uniform in τ. We use the fact that theta constants converge quickly as the smallest eigenvalue $\lambda_1(\tau)$ of $\mathrm{Im}(\tau)$ tends to infinity. Moreover, $\lambda_1(\gamma_i\tau)$ is bounded from below when $\tau \in \mathcal{F}_2$ and $|\tau|$ is not too large.

Lemma 5.3. *For every $\tau \in \mathcal{H}_2$ such that $\lambda_1(\tau) \geq 1$, we have*

$$|\theta_j(\tau) - 1| < 4.18 \exp(-\pi\lambda_1(\tau)) \qquad \text{for } 0 \leq j \leq 3.$$

Proof. Let $0 \leq j \leq 3$. Using the series expansion of θ_j, we obtain

$$|\theta_j(\tau) - 1| \leq \sum_{n \in \mathbb{Z}^2 \setminus \{0\}} \exp(-\pi n^t \,\mathrm{Im}(\tau) n) \leq \sum_{n \in \mathbb{Z}^2 \setminus \{0\}} \exp(-\pi\lambda_1(\tau)\,\|n\|^2).$$

By splitting the plane into quadrants, we see that this last sum is equal to $4S^2 + 4S$, with

$$S = \sum_{n \geq 1} \exp(-\pi\lambda_1(\tau)n^2) \leq \frac{\exp(-\pi\lambda_1(\tau))}{1 - \exp(-3\pi\lambda_1(\tau))}.$$

Since $\lambda_1(\tau) \geq 1$, the conclusion follows. □

Lemma 5.4. *Let $\tau \in \mathcal{H}_2$ and $\gamma \in \mathrm{Sp}_4(\mathbb{Z})$. Then*

$$\lambda_1(\gamma\tau) \geq \frac{\det \mathrm{Im}(\tau)}{8\,|\gamma|^2\,|\tau|\,(2\,|\tau| + 1)^2}.$$

Proof. We have

$$\lambda_1(\gamma\tau) \geq \frac{\det \mathrm{Im}(\gamma\tau)}{\mathrm{Tr}\,\mathrm{Im}(\gamma\tau)}.$$

By [20, (5.11) p. 57],

$$\mathrm{Im}(\gamma\tau) = (\gamma^*\tau)^{-t}\,\mathrm{Im}(\tau)(\gamma^*\bar{\tau})^{-1}$$

so that

$$\det \mathrm{Im}(\gamma\tau) = \frac{\det \mathrm{Im}(\tau)}{|\det(\gamma^*\tau)|^2},$$

$$\mathrm{Tr}\,\mathrm{Im}(\gamma\tau) \leq 8\left|(\gamma^*\tau)^{-1}\right|^2 |\mathrm{Im}(\tau)| \leq 8\frac{|\gamma^*\tau|^2\,|\tau|}{|\det(\gamma^*\tau)|^2} \leq 8\frac{|\gamma|^2\,(2\,|\tau| + 1)^2\,|\tau|}{|\det(\gamma^*\tau)|^2}.$$

The result follows. □

Proposition 5.5. *There is an algorithm such that the following holds. Let $\tau \in \mathcal{F}_2$ and $N \geq 1$. Then, given approximations of squares of theta quotients at τ to precision N, the algorithm computes an approximation of τ within*

$$O\big(\mathcal{M}(N + |\tau|) \log |\tau| + \mathcal{M}(N) \log N\big)$$

binary operations. The precision loss is $O(\log N + |\tau| \log |\tau|)$ bits.

Proof. We obtain the quantities

$$(\theta_j^2(2^n \gamma_i \tau)/\theta_0^2(\gamma_i \tau))_{0 \leq j \leq 3}$$

after n Borchardt steps. By Lemma 5.4, we know that

$$|\log \lambda_1(\gamma_i \tau)| = O(\log |\tau|).$$

Therefore, we can choose $n = O(\log |\tau|)$ such that $\lambda_1(2^n \gamma_i \tau) \geq 10$, for instance. Up to this point, we performed $O(\log |\tau|)$ elementary operations with complex numbers z such that $\log |z| = O(|\tau|)$. Therefore the total cost is

$$O(\mathcal{M}(N + |\tau|) \log |\tau|)$$

binary operations, and the precision loss is $O(|\tau| \log |\tau|)$ bits. Even if $|\tau|$ is not known explicitly, this moment is detected in the algorithm as when the four values in the Borchardt sequence get very close to each other.

Then, we normalize so that one of the four values is 1, and continue performing a further $O(\log N)$ Borchardt steps: this O-constant and the accuracy of the result can be made explicit by [4, Prop. 7.2]. This costs $O(\mathcal{M}(N) \log N)$ binary operations, and the precision loss is $O(\log N)$ bits. This allows us to compute the quantities $\theta_0^2(\gamma_i \tau)$ for $1 \leq i \leq 3$; the precision loss up to now is $O(\log N + |\tau| \log |\tau|)$ bits.

Finally, we recover the coordinates of τ using Eq. (7). This final computation costs $O(N + |\tau|)$ binary operations, and the precision loss is $O(|\tau|)$ bits. This ends the proof. □

Proposition 5.6. *There exists an algorithm such that the following holds. Let $j_1, j_2, j_3 \in F$, let σ be a complex embedding of F, and choose B_σ as above. Let $N \geq 1$. Then, given approximations of $\sigma(j_k)$ for $1 \leq k \leq 3$, the algorithm computes $\tau \in \mathcal{F}_2$ such that the Igusa invariants at τ are the $\sigma(j_k)$ for $1 \leq k \leq 3$. This algorithm involves $O\big(\mathcal{M}(N + B_\sigma) \log(N + B_\sigma)\big)$ binary operations, and a precision loss of $O(\log N + B_\sigma \log B_\sigma)$ bits.*

Proof. First, we compute a curve $\mathcal{C}$ as in Lemma 5.1. Then, by Thomae's formula [16, Thm. IIIa.8.1], there is a finite number of possibilities for the values of squares of theta quotients at τ; one of them corresponds to an actual $\tau \in \mathcal{F}_2$, and the others correspond to other elements in the orbit $\Gamma(1)\tau$. When we run the algorithm of Proposition 5.5 on these inputs, we may discard all resulting period matrices that do not belong to $\mathcal{F}_2$. In order to distinguish between the remaining possible values of τ, it is usually enough to compute theta constants

to precision $O(1)$ using the naive algorithm, and match with the input. In extreme cases, we may resort to computing Igusa invariants at all remaining possible values of τ to precision $O(N+B_\sigma)$, using $O\big(\mathcal{M}(N+B_\sigma)\log(N+B_\sigma)\big)$ binary operations, by Proposition 4.3. □

6 Evaluating Hilbert modular polynomials

In this final section, we give the complete algorithm to evaluate Hilbert modular polynomials of level β for $K=\mathbb{Q}(\sqrt{5})$, given Gundlach invariants (g_1,g_2) in a number field F. Let (j_1,j_2,j_3) be the associated Igusa invariants. In every complex embedding σ of F, we compute a period matrix $\tau\in\mathcal{F}_2$ with Igusa invariants (j_1,j_2,j_3), and invert the Hilbert embedding to find $t\in\mathcal{H}_1^2$ such that $\Phi_K(t)$ is $\Gamma(1)$-equivalent to τ. Then we evaluate the numerators and denominators of Hilbert modular polynomials of level β, by computing theta constants at $\Phi_K(\frac{1}{\beta}\eta t)$ for $\eta\in C_\beta^\sigma$. Finally, we recognize the coefficients as algebraic integers.

The case of Siegel modular polynomials is simpler, since we do not have to invert the Hilbert embedding. We do not detail it and simply point out the differences in running time.

6.1 Inverting the Hilbert embedding

Recall that for every $\tau\in\mathcal{H}_2$, we denote by $\lambda_1(\tau)\le\lambda_2(\tau)$ the two eigenvalues of $\mathrm{Im}(\tau)$.

Lemma 6.1. *Let K be a real quadratic field. Then there exists a constant $C>0$ depending only on the choice of Hilbert embedding such that for every $t=(t_1,t_2)\in\mathcal{H}_1^2$, we have*

$$\tfrac{1}{C}\lambda_1(\Phi_K(t))\le\min\{\mathrm{Im}(t_1),\mathrm{Im}(t_2)\}\le C\lambda_1(\Phi_K(t)),$$
$$\tfrac{1}{C}\lambda_2(\Phi_K(t))\le\max\{\mathrm{Im}(t_1),\mathrm{Im}(t_2)\}\le C\lambda_2(\Phi_K(t)).$$

Proof. Use formula (3) describing the Hilbert embedding. □

Proposition 6.2. *There exists an algorithm and a constant C depending only on K such that the following holds. Let $j_1,j_2,j_3\in F$ such that $j_3\neq 0$, let σ be a complex embedding of F, and define B_σ as in §4. Let $\mathcal{C}$ be a genus 2 hyperelliptic curve over $\mathbb{C}$ with Igusa invariants $\sigma(j_1),\sigma(j_2),\sigma(j_3)$, let $\tau\in\mathcal{F}_2$ be its period matrix, and assume that $\mathrm{Jac}(\mathcal{C})$ has real multiplication by $\mathbb{Z}_K$. Then there exists $t=(t_1,t_2)\in\mathcal{H}_1^2$ such that $\Phi_K(t)$ is a period matrix of $\mathcal{C}$, and*

$$|\log(\mathrm{Im}\,t_i)|\le CB_\sigma\quad\text{for } i=1,2.$$

Moreover, given an approximation of τ to precision $N+CB_\sigma$, the algorithm computes an approximation of t at precision N within $O_K(\mathcal{M}(N+B_\sigma)\log B_\sigma)$ binary operations.

The notation O_K indicates that the implied constant depends on K.

Proof. By Lemma 5.2, there is an absolute constant C such that

$$|\tau| \leq CB_\sigma.$$

The result would be obvious from Lemma 6.1 if there existed $t \in \mathcal{H}_1^2$ such that $\tau = \Phi_K(t)$, but this is not always the case. In general, by [2, Lem. 4.1], we know that there are coprime integers a, b, c, d, e such that

$$b^2 - 4ac - 4de = \Delta_K \quad \text{and} \quad a\tau_1 + b\tau_3 + c\tau_2 + d\det(\tau) + e = 0. \tag{8}$$

We claim that the heights of a, b, c, d, e must be in $O_K(B_\sigma)$. Looking at the complex torus

$$A_\tau = \mathbb{C}^2/(\tau\mathbb{Z}^2 \oplus \mathbb{Z}^2),$$

the rational representation of an endomorphism f in the image of $\mathbb{Z}_K$ inside $\mathrm{End}(A_\tau)$ is of the form

$$\rho_{R,\tau}(f) = \begin{pmatrix} n & ma & 0 & md \\ -mc & mb+n & -md & 0 \\ 0 & me & n & -mc \\ -me & 0 & ma & mb+n \end{pmatrix} \qquad \text{for some } m, n \in \mathbb{Z}.$$

by [2, Cor. 4.2]. On the other hand, the analytic representation $\rho_{A,\tau}(\sqrt{\Delta})$ of the endomorphism $\sqrt{\Delta}$ of A_τ can be computed as follows. Let $\omega = (\omega_1, \omega_2)$ be a basis of differential forms on A_τ such that $\mathrm{Sym}^2(\omega)$ corresponds by the Kodaira–Spencer isomorphism to a deformation of A_τ along the Humbert surface. Then, the analytic representation of $\sqrt{\Delta}$ in the basis ω is of the form

$$\pm \begin{pmatrix} \sqrt{\Delta} & 0 \\ 0 & -\sqrt{\Delta} \end{pmatrix}.$$

Algorithm 4.6 from [11] shows that such a basis ω exists; moreover the base change matrix m between (dz_1, dz_2) and ω can be chosen such that

$$\log \max\{|m|, |m^{-1}|\} = O_K(B_\sigma).$$

This proves that the analytic representation of the endomorphism $\sqrt{\Delta}$ on A_τ satisfies

$$\log^+ \left|\rho_{A,\tau}(\sqrt{\Delta})\right| = O_K(B_\sigma).$$

For every $f \in \mathrm{End}(A_\tau)$, the rational and analytic representations of f are related by the formula [1, Rem. 8.14]

$$\rho_{A,\tau}(f)(\tau\ I_2) = (\tau\ I_2)\rho_{R,\tau}(f).$$

Taking imaginary parts, we find that there exist $m, n \in \mathbb{Z}$ such that

$$\operatorname{Im}(\tau)\begin{pmatrix} n & ma \\ -mc & mb+n \end{pmatrix} = \operatorname{Im}\big(\rho_{A,\tau}(\sqrt{\Delta})\tau\big),$$
$$\operatorname{Im}(\tau)\begin{pmatrix} 0 & md \\ -md & 0 \end{pmatrix} = \operatorname{Im}\big(\rho_{A,\tau}(\sqrt{\Delta})\big).$$

Therefore a, b, c, d, m, n must be bounded by $O_K(B_\sigma)$ in height. The same is true for e by Eq. (8). This proves our claim.

The algorithm to compute t is as follows. We compute the integers a, b, c, d, e in $O_K(\mathcal{M}(B_\sigma)\log B_\sigma)$ binary operations with the LLL algorithm, using Eq. (8). Then, we use the algorithm from [2, Prop. 4.5] to compute a matrix $\gamma \in \Gamma(1)$ such that $\gamma\tau$ lies in the image of Φ_K; the matrix γ has a simple expression in terms of a, b, c, d, e, hence we also have

$$\log|\gamma| = O_K(B_\sigma).$$

By Lemma 5.4, we also have

$$\Lambda(\gamma\tau) \in O_K(B_\sigma),$$

so the result follows from Lemma 6.1. □

6.2 Analytic evaluation of modular polynomials

Fix $K = \mathbb{Q}(\sqrt{5})$. Let F be a number field, let σ be a complex embedding of F, and let $(g_1, g_2) \in F$. Choose β and ℓ as in §3.2, and define B_σ as in §4. In the following proposition, we detail the algorithm to evaluate the numerator and denominator of Hilbert modular polynomials at $(\sigma(g_1), \sigma(g_2))$. We may assume that $g_1 \neq 0$; otherwise, the denominator of modular polynomials vanishes. In order to avoid complicated expressions, we hide logarithmic factors in the $\widetilde{O}$ notation from now on. Actually $\widetilde{O}(T)$ will always be $O(\mathcal{M}(T\log T)\log T)$.

Proposition 6.3. *Under hypothesis 4.1, there exists an algorithm and a constant C such that the following holds. Let $N \geq 1$. Then, given approximations of $\sigma(g_1)$ and $\sigma(g_2)$ to precision N, the algorithm computes $\sigma(D_\beta(g_1, g_2))$ and $\sigma(D_\beta\Psi_{\beta,k}(g_1, g_2))$ for $k \in \{1, 2\}$ within $\widetilde{O}(\ell B_\sigma^2 + \ell N)$ binary operations, with a precision loss of $\widetilde{O}(\ell B_\sigma + \log N)$ bits.*

Proof. We first compute the associated Igusa invariants $\sigma(j_k)$ for $1 \leq k \leq 3$ using [19, Thm. 1]; see also [15, Thm. 2.13]. Note that $j_3 \neq 0$. Using Proposition 5.5, we compute a period matrix $\tau \in \mathcal{F}_2$ having these Igusa invariants in $\widetilde{O}(N + B_\sigma)$ binary operations, with a precision loss of $O(\log N + B_\sigma \log B_\sigma)$ bits. Then, using Proposition 6.2, we compute $t \in \mathcal{H}_1^2$ such that t has Gundlach invariants (g_1, g_2), and

$$|\log(\operatorname{Im} t_i)| = O_K(B_\sigma), \quad i \in \{1, 2\}.$$

This costs $\widetilde{O}(N+B_\sigma)$ binary operations, with a precision loss of $O(B_\sigma)$ bits. The next step is to compute the quantities $\frac{1}{\beta}\eta t$ for $\eta \in C_\beta^\sigma$: this costs $\widetilde{O}(\ell(N+B_\sigma))$ binary operations, with a precision loss of $O(\log \ell)$ bits. By Lemma 6.1, we have for every $\eta \in C_\beta^\sigma$:

$$\Lambda(\Phi_K(\tfrac{1}{\beta}\eta t)) = O(B_\sigma + \log \ell).$$

We reduce the matrices $\Phi_K(\frac{1}{\beta}\eta t)$ and compute $\tau_\eta \in \mathcal{F}_2$ and $\gamma_\eta \in \Gamma(1)$ such that

$$\gamma_\eta(\tfrac{1}{\beta}\eta t) = \tau_\eta, \qquad \text{for every } \eta \in C_\beta^\sigma.$$

We also compute squares of theta constants at τ and every τ_η. By Corollary 4.9, this can be done within

$$\widetilde{O}(\ell B_\sigma^2 + \ell N)$$

binary operations, with a precision loss of $O(B_\sigma + \log \ell)$ bits. Moreover, we have

$$\log |\gamma_\eta| \in O(B_\sigma + \log \ell).$$

This yields the values of h_4, h_6, h_{10}, h_{12} at the matrices τ_η using $O(\ell)$ binary operations, and a precision loss of $O(1)$ bits.

At the end, we evaluate $D_\beta(t)$ using Eq. (4), and the equality

$$h_{10}^2(\tfrac{1}{\beta}\eta t) = (\det \gamma_\eta^*(\tfrac{1}{\beta}\eta t))^{-20} h_{10}^2(\tau_\eta),$$

for every $\eta \in C_\beta^\sigma$. By Lemma 4.6, the total precision loss in this computation is $O(\ell(B_\sigma + \log \ell))$; the total cost of computing $D_\beta(t)$ is $\widetilde{O}(\ell(N+B_\sigma))$ binary operations. Up to a similar scalar factor, the polynomials $\sigma(\Psi_{\beta,k}(g_1, g_2))$ for $k \in \{1, 2\}$ are given by

$$\prod_{\eta \in C_\beta^\sigma} (F_{10}(\tau_\eta)X + G_2^5(\tau_\eta)) \qquad \text{for } k = 1,$$

$$\sum_{\eta \in C_\beta^\sigma} G_2^5(\tau_\eta) \prod_{\eta' \in C_\beta^\sigma \setminus \{\eta\}} (F_{10}(\tau_{\eta'})X - G_2^2(\tau_{\eta'})F_6(\tau_{\eta'})) \qquad \text{for } k = 2.$$

By Lemma 2.3, these polynomials can be computed in $\widetilde{O}(\ell N)$ binary operations, with a precision loss of $O(\ell)$ bits. We conclude by summing precision losses and binary costs of each step. □

In the case of Siegel modular polynomials, the complexity and precision loss estimates are similar, with each occurence of ℓ replaced by ℓ^3.

6.3 Algebraic evaluation of modular polynomials

Once modular polynomials and their denominators have been computed in every complex embedding, we only have to recognize their coefficients as algebraic numbers. We present two results, one in the case of a finite field, and the second in the case of a number field. Recall that ℓ is a prime that splits in

$K = \mathbb{Q}(\sqrt{5})$ into two principal ideals generated by totally positive elements, β and $\overline{\beta}$.

In the case of a finite field, we are given a prime power $q = p^d$, and a monic polynomial $P \in \mathbb{Z}[X]$ of degree d, irreducible modulo p. We let $M \geq 1$ such that $\log |P| \leq M$. We assume that a black box provides us with approximations of the roots of P to any desired precision. Then, we represent elements of $\mathbb{F}_q$ as elements of $\mathbb{F}_p[X]/(P)$.

Proposition 6.4. *Under the conditions of the previous paragraph and under hypothesis 4.1, there exists an algorithm such that the following holds: given ℓ and $g_1, g_2 \in \mathbb{F}_q$ such that $D_\ell(g_1, g_2) \neq 0$, the algorithm computes the polynomials $\Psi_{\beta,k}(g_1, g_2) \in \mathbb{F}_q[X]$ for $k \in \{1, 2\}$ within*

$$\widetilde{O}(\ell d^2 \log^2 p + \ell d^4 M^2 + \ell^2 d \log p + \ell^2 d^2 M)$$

binary operations.

If $dM = O(\log p)$, and if moreover $\ell = O(\log q)$, then the cost estimate simplifies to $\widetilde{O}(\log^3 q)$ binary operations. If $q = p$ is prime (i.e. $d = 1$), then the cost estimate simplifies to $\widetilde{O}(\ell \log^2 p + \ell^2 \log p)$ binary operations. Theorem 1.1 stated in the introduction is the analogue of Proposition 6.4 for Siegel modular polynomials, where we replace ℓ by ℓ^3.

Proof. Let F be the number field $\mathbb{Q}[X]/(P)$, and let α be a root of P in F. We lift g_1, g_2 to elements of $\mathbb{Z}[\alpha]$, with coefficients bounded by $\log p$ in height. Then

$$\begin{aligned} h(\alpha) &\leq M + \log 2, \\ \max\{h(g_1), h(g_2)\} &\leq \log(p) + dh(\alpha) + \log(d) = O(dM + \log p). \end{aligned}$$

Since D_β and the coefficients of $\Psi_{\beta,k}$ are polynomials in $\mathbb{Z}[g_1, g_2]$ of degree $O(\ell)$ and height $O(\ell \log \ell)$, the algebraic integers we have to recognize all are elements of $\mathbb{Z}[\alpha]$, with coefficients bounded by $O(\ell \log \ell + \ell dM + \ell \log p)$ in height. By Lemma 2.5, we can recognize each coefficient within $\widetilde{O}(\ell d^2 M + \ell d \log p)$ binary operations, provided that its values in every complex embedding of F are computed to precision at least $C(\ell \log \ell + \ell dM + \ell \log p)$ in every complex embedding of F, where C is some absolute constant.

Let σ be a complex embedding of F, and start at precision $N \geq 1$. Then $\sigma(g_1)$, $\sigma(g_2)$ are obtained by replacing α by one of the complex roots of P: this can be done within $\widetilde{O}(d(M + N))$ binary operations, and a precision loss of $O(dM + \log p)$ bits, via Horner's algorithm. Then we run the algorithm of Proposition 6.3, for each complex embedding σ of F. It is enough to choose N in

$$\widetilde{O}(\ell dM + \ell \log p + \ell B_\sigma),$$

for a cost of $\widetilde{O}(\ell B_\sigma^2 + \ell N)$ binary operations. Since we have

$$\sum_\sigma B_\sigma = O(d \log p + d^2 M),$$

and each B_σ is in $O(d \log p + d^2 M)$, the total cost of analytic evaluations over all embeddings is

$$\widetilde{O}(\ell d^2 \log^2 p + \ell d^4 M^2 + \ell^2 d \log p + \ell^2 d^2 M)$$

binary operations, and dominates the cost of algebraic reconstruction. □

If $g_1, g_2 \in \mathbb{Z}$ are small integers, then the complexity of evaluating modular equations is quasi-linear in the output size.

Proposition 6.5. *There exists an algorithm such that the following holds. Given the prime ℓ and $g_1, g_2 \in \mathbb{Z}$ such that*

$$\max\{|g_1|, |g_2|\} \in O(1) \quad \textit{and} \quad D_\beta(g_1, g_2) \neq 0 \bmod p,$$

the algorithm computes the polynomials $\Psi_{\beta,k}(g_1, g_2) \in \mathbb{Q}[X]$ for $k \in \{1, 2\}$ within $O(\mathcal{M}(\ell^2 \log \ell) \log \ell)$ binary operations.

Proof. In this case, we have $B_\sigma = O(1)$. It is sufficient to round the result of Proposition 6.3 with $N = C\ell \log \ell$, where C is an absolute constant, to the nearest integers. □

From another point of view, the complexity of evaluating Hilbert modular polynomials over number fields can be bounded in terms of the discriminant and the height of the operands. We assume that an LLL-reduced integer basis of the number field F has been precomputed. Moreover, if m_F is the matrix defined in Lemma 2.6, we assume that a black box provides us with the coefficients of m_F^{-1} to any desired precision.

Proposition 6.6. *Under the conditions from the previous paragraph and hypothesis 4.1, there exists an algorithm such that the following holds. Let $H \geq 1$, and let $g_1, g_2 \in F$ given as quotients of integers of height at most H such that $D_\beta(g_1, g_2) \neq 0$. Then the algorithm computes $\Psi_{\beta,k}(g_1, g_2) \in F[X]$ for $k \in \{1, 2\}$ within*

$$\widetilde{O}(d^2 \ell \log \Delta_F + d^3 \ell + \ell d^2 H^2 + \ell^2 d^2 H)$$

binary operations.

In the case $F = \mathbb{Q}$, the cost estimate simplifies to $\widetilde{O}(\ell H^2 + \ell^2 H)$ binary operations.

Proof. For simplicity, assume that g_1 and g_2 are actually integers: in the general case we multiply D_β by an appropriate power of a common denominator of g_1 and g_2 in $\mathbb{Z}_F$.

We know that $D_\beta(g_1, g_2)$ and the coefficients of $D_\beta \Psi_{\beta,k}(g_1, g_2)$ are polynomials in $\mathbb{Z}[g_1, g_2]$ of degree $O(\ell)$ and height $O(\ell \log \ell)$: hence they are algebraic integers of height $\widetilde{O}(\ell H)$. By Lemma 2.6, we can recognize each coefficient within $\widetilde{O}(d^2 \ell H + d^2 \log \Delta_F + d^3)$ binary operations, provided that complex approximations are computed at a precision N high enough; it is enough to take N in $\widetilde{O}(\log \Delta_F + d\ell H)$.

In order to obtain these approximations, we run the algorithm of Proposition 6.3 in each complex embedding σ of F, at precision. For each σ, the starting precision is chosen in $\widetilde{O}(\log \Delta_F + d\ell H + \ell B_\sigma)$, so the cost to compute the complex approximation in the embedding σ is $\widetilde{O}(\ell B_\sigma^2 + \ell \log \Delta_F + d\ell^2 H + \ell^2 B_\sigma)$ binary operations. The sum of the bounds B_σ is in $O(dH)$, as well as each individual B_σ. Therefore, the total cost for all embeddings is $\widetilde{O}(\ell d^2 H^2 + \ell d \log \Delta_F + \ell^2 d^2 H)$ binary operations. □

The complexity results of Propositions 6.4 and 6.6 are not entirely satisfactory: the dependence on $\log p$ in the finite field case, and on H in the number field case, is quadratic. This comes from the fact that the reduction algorithm to the Siegel fundamental domain (Proposition 4.8) is quasi-quadratic in $\Lambda(\tau)$. Reduction to the fundamental domain in genus 2 is essentially equivalent to lattice reduction for dimension 4 symplectic lattices, and there are known instances where lattice reduction can be performed in quasi-linear time [18], so there is certainly some room left for improvement.

References

[1] C. Birkenhake and H. Lange. *Complex abelian varieties*. Springer-Verlag, Berlin, second edition, 2004.

[2] C. Birkenhake and H. Wilhelm. Humbert surfaces and the Kummer plane. *Trans. Amer. Math. Soc.*, 335(5):1819–1841, 2003.

[3] A. Bostan, F. Chyzak, M. Giusti, R. Lebreton, G. Lecerf, B. Salvy, and E. Schost. *Algorithmes efficaces en calcul formel*. Printed by CreateSpace, 2017.

[4] R. Dupont. *Moyenne arithmético-géométrique, suites de Borchardt et applications*. PhD thesis, École polytechnique, 2006.

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

[6] J.-I. Igusa. *Theta functions*. Springer-Verlag, 1972.

[7] J.-i. Igusa. On the ring of modular forms of degree two over $\mathbb{Z}$. *Amer. J. Math.*, 101(1):149–183, 1979.

[8] F. Johansson. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Trans. Comput.*, 66(8):1281–1292, 2017.

[9] J. Kieffer. Degree and height estimates for modular equations on PEL Shimura varieties. 2020.

[10] J. Kieffer. Sign choices in the AGM for genus two theta constants. 2020.

[11] J. Kieffer, A. Page, and D. Robert. Computing isogenies from modular equations in genus two. 2019.

[12] C. Martindale. Hilbert modular polynomials. *J. Number Theory*, 213:464–498, 2020.

[13] J.-F. Mestre. Construction de courbes de genre 2 à partir de leurs modules. In *Effective methods in algebraic geometry (Castiglioncello, 1990)*, volume 94 of *Progr. Math.*, page 313–334. Birkhäuser, Boston, 1991.

[14] E. Milio. A quasi-linear time algorithm for computing modular polynomials in dimension 2. *LMS J. Comput. Math.*, 18:603–632, 2015.

[15] E. Milio and D. Robert. Modular polynomials on Hilbert surfaces. *J. Number Theory*, 2020.

[16] D. Mumford. *Tata lectures on theta. II.* Number 43 in Progr. Math. Birkhäuser, Boston, 1984.

[17] S. Nagaoka. On the ring of Hilbert modular forms over $\mathbb{Z}$. *J. Math. Soc. Japan*, 35(4):589–608, 1983.

[18] A. Novocin, D. Stehlé, and G. Villard. An LLL-reduction algorithm with quasi-linear time complexity. In *STOC'11 - 43rd annual ACM symposium on Theory of computing*, pages 403–412, San Jose, United States, 2011.

[19] H. L. Resnikoff. On the graded ring of Hilbert modular forms associated with $\mathbb{Q}(\sqrt{5})$. *Math. Ann.*, 208:161–170, 1974.

[20] M. Streng. *Complex multiplication of abelian surfaces.* PhD thesis, Universiteit Leiden, 2010.