HAL open science

The complexity of class polynomial computation via floating point approximations

Andreas Enge

To cite this version:

Andreas Enge. The complexity of class polynomial computation via floating point approximations. 2006. inria-00001040v1

HAL Id: inria-00001040

https://inria.hal.science/inria-00001040v1

Submitted on 24 Jan 2006 (v1), last revised 25 Jul 2008 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

inria-00001040, version 1 - 24 Jan 2006

The complexity of class polynomial computation via floating point approximations

Andreas Enge
INRIA Futurs & Laboratoire d'Informatique (CNRS/UMR 7161)
École polytechnique, 91128 Palaiseau Cedex, France
enge@lix.polytechnique.fr

January 18, 2006

Abstract

We analyse the complexity of computing class polynomials, that are an important ingredient for CM constructions of elliptic curves, via complex floating point approximations of their roots. The heart of the algorithm is the evaluation of modular functions in several arguments. The fastest one of the presented approaches uses a technique devised by Dupont to evaluate modular functions by Newton iterations on an expression involving the arithmetic-geometric mean. It runs in time

$$O\left(|D| \log^5 |D| \log\log |D|\right) = O\left(|D|^{1+\varepsilon}\right) = O\left(h^{2+\varepsilon}\right)$$

for any $\varepsilon > 0$, where D is the CM discriminant and h is the degree of the class polynomial. Another fast algorithm uses multipoint evaluation techniques known from symbolic computation; its asymptotic complexity is worse by a factor of $\log |D|$. Up to logarithmic factors, this running time matches the size of the constructed polynomials. The estimate also relies on a new result concerning the complexity of enumerating the class group of an imaginary-quadratic order and on a rigorously proven asymptotic upper bound for the height of class polynomials, that matches experimental findings.

1 Motivation and results

The theory of complex multiplication yields an efficient approach to the construction of elliptic curves over a finite field having a given endomorphism ring, as long as the absolute value of its discriminant D is not too large. Exploiting the link between the endomorphism ring of an elliptic curve and its number of points, it is possible to obtain efficiently curves with specific properties. Applications include primality proving [2], the construction of classical, discrete logarithm based elliptic curve cryptosystems and of identity based cryptosystems [32, 16, 3, 5].

As indicated by the name, the starting point for complex multiplication is complex analysis. More precisely, the usual approach relies on the computation of minimal polynomials of special, algebraic values of modular functions on the upper complex halfplane (more details are provided in Section 2). The complexity of this algorithm remains a shady issue; one of the reasons a serious analysis has not been undertaken so far are the purported numerical difficulties with the computations with complex floating point numbers, an issue that actually does

not arise in practice (cf. the short discussion at the end of Section 7). Moreover, figures found in the existing literature need not necessarily reflect reality. For instance, in [1] it is claimed in the introduction that the floating point approach has a complexity of $O(|D|^2 \log^2 |D|)$, while even the most naïve algorithm presented in Section 5.1 in fact has a complexity of only $O(|D|^{3/2+\varepsilon})$. The authors of [8] even pretend that the complexity is exponential in the class number h and thus in $|D|$!

In [10] the authors present an algebraic generalisation of the complex multiplication algorithms to p-adic fields, that yields the same minimal polynomials. Besides ruling out any possible numerical instabilities, they obtain a complexity of $O(|D|^{1+\varepsilon})$, which is better than a straightforward implementation of the complex approach, and asymptotically optimal (up to logarithmic factors) due to the size of the constructed objects.

The present article thus has two goals. First, it provides an accurate account of the complexity of different algorithms for class polynomial computation via floating point approximations. Second, it shows that asymptotically optimal algorithms exist also in the complex setting, with a complexity that is linear (up to logarithmic factors) in the output size. These new algorithms are presented in Sections 5.3 and 5.4, and the faster one achieves the following:

Theorem 1
Let f be a fixed modular function that is a class invariant for a family of discriminants D of class numbers $h = h(D)$. Assume that all floating point computations are carried out with $n = n(D)$ bits, and that this precision suffices to neutralise possible rounding errors. Then there is a deterministic algorithm that computes the class polynomials for f in time

$$O\left(hn \log^2 n \log\log n\right).$$

Taking $n = O(\sqrt{|D|} \log^2 |D|)$ according to Theorem 2 and using the bound on the class number $h = O(|D| \log |D|)$ mentioned at the end of Section 6, the complexity becomes

$$O\left(|D| \log^5 |D| \log\log |D|\right) = O\left(|D|^{1+\varepsilon}\right) = O\left(h^{2+\varepsilon}\right)$$

for any $\varepsilon > 0$.

Notice that up to logarithmic factors, this complexity corresponds to the output of the algorithm, namely the size of the class polynomials.

The floating point precision n required to carry out the computations depends crucially on the height of the class polynomials; assuming that rounding errors do not perturb the situation, both are actually the same. The following theorem provides a rigorously proven upper bound on the heights, that is close to experimental findings. An upper bound of the same shape is given in [1, §3.3] as a heuristic, and the gist of the proof appears already in [29, §5.10].

Theorem 2
The logarithmic height of the class polynomial for j for the discriminant D of class number h is bounded above by

$$\log(2c')h + \pi\sqrt{|D|} \log^2 |D| + O\left(\sqrt{|D|} \log |D|\right) = O\left(\sqrt{|D|} \log^2 |D|\right)$$

with $c' = 5.939\ldots$. The bound of

$$O\left(\sqrt{|D|}\log^2|D|\right)$$

holds for any other class invariant as well.

As a byproduct of the proof of Theorem 1 we obtain in Section 4 the following result for computing class groups of imaginary-quadratic orders:

Theorem 3
There is a deterministic algorithm that enumerates the class group of the imaginary-quadratic order of discriminant D and class number h in time

$$O\left(|D|^{3/4+\varepsilon}\right) = O\left(h^{3/2+\varepsilon}\right)$$

for any $\varepsilon > 0$.
With a randomised algorithm, the same goal may be achieved in time

$$O\left(|D|^{1/2+\varepsilon}\right) = O\left(h^{1+\varepsilon}\right).$$

Again this algorithm is essentially optimal in its output size.

2 Complex multiplication and class polynomials

2.1 The basic approach

For proofs of the following facts on complex multiplication of elliptic curves see, for instance, [11].

Let us first consider the situation over the complex numbers. Let $D < 0$ be an imaginary-quadratic discriminant, and $\mathcal{O}_D = \left[1, \frac{D+\sqrt{D}}{2}\right]_{\mathbb{Z}}$ the (not necessarily maximal) order of discriminant D in $K = \mathbb{Q}(\sqrt{D})$. The ideal class number of $\mathcal{O}_D$ is denoted by $h = h_D$. By Siegel's theorem [36], $\frac{\log h}{\log|D|} \to \frac{1}{2}$ ($|D| \to \infty$), so that $|D| \in O(h^{2+\varepsilon})$ and $h \in O(|D|^{1/2+\varepsilon})$ for any $\varepsilon > 0$. There are h non-isomorphic elliptic curves over $\mathbb{C}$ having*complex multiplication* by $\mathcal{O}_D$, that is, $\mathcal{O}_D$ as endomorphism ring. Namely, letting $j : \mathbb{H} = \{z \in \mathbb{C} : \Im(z) > 0\} \to \mathbb{C}$ denote the absolute modular invariant, and letting $\tau_i = \frac{-B_i+\sqrt{D}}{2A_i}$ run through the roots of the reduced quadratic forms $[A_i, B_i, C_i] = A_iX^2 + B_iX + C_i$ of discriminant $D = B_i^2 - 4A_iC_i$, representing the ideal classes of $\mathcal{O}_D$, then the j-invariants of the elliptic curves are given by the $j(\tau_i)$. Moreover, these $j(\tau_i)$ are algebraic integers; in fact, they generate the so-called*ring class field* K_D for $\mathcal{O}_D$, the Galois extension of K whose Galois group is precisely the class group of $\mathcal{O}_D$. The minimal polynomial of the $j(\tau_i)$, $H_D(X) = \prod_{i=1}^{h}(X - j(\tau_i))$, has coefficients in $\mathbb{Z}$ and is called a*class polynomial*. In the special case that D is a fundamental discriminant, the ring class field K_D is also called the*Hilbert class field* of K.

Let now $\mathbb{F}_q = \mathbb{F}_{p^m}$ be a finite field of characteristic p. If q may be written as $4q = U^2 + DV^2$ with U, $V \in \mathbb{Z}$ and $p \nmid D$, then p splits in $K = \mathbb{Q}(\sqrt{D})$ and is unramified in K_D, and the prime ideals above p in K_D are of inertia degree m. The reductions of the complex elliptic curves with complex multiplication by $\mathcal{O}_D$

modulo any of these prime ideals thus live in $\mathbb{F}_q$, and by Deuring's reduction theorem [13], they still have complex multiplication by $\mathcal{O}_D$. Moreover, these h curves are precisely the curves with complex multiplication by $\mathcal{O}_D$ over $\mathbb{F}_q$. This observation suggests a simple algorithm: compute the class polynomial $H_D \in \mathbb{Z}[X]$ and reduce it modulo p. It splits completely over $\mathbb{F}_q$, and each of its roots is the j-invariant of a looked for elliptic curve.

2.2 Class invariants

Unfortunately, H_D has very large coefficients (see the discussion in Section 6), so that its computation requires a high precision to be accurate. To remedy to this situation, it is possible to use different modular functions f in the place of j. These functions are invariant under $\Gamma^0(N) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{Sl}_2(\mathbb{Z}) : N|b \right\}$ for some positive integer N, that is, $f\left(\frac{az+b}{cz+d}\right) = f(z)$ for any such matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Under suitable conditions on the discriminant D and suitable normalisations of the τ_i, derived from Shimura's reciprocity law, the*singular values* $f(\tau_i)$ are still elements of the class field K_D [34, 25, 26, 35]; we then call f a*class invariant* and the minimal polynomial $H_D[f](X) = \prod_{i=1}^{h}(X - f(\tau_i))$ again a*class polynomial*.

The first such class invariants are given by Weber's functions $\mathfrak{f}$, $\mathfrak{f}_1$, $\mathfrak{f}_2$, γ_2 and γ_3 [38].

Two parameterised families of class invariants are exhibited in [19], where N is the product of two primes and $H_D[f] \in \mathbb{Z}[X]$, and [18], where N is prime and $H_D[f]$ has coefficients in the maximal order of $\mathbb{Q}(\sqrt{D})$. Again, these class polynomials split completely after reduction in $\mathbb{F}_q$. The corresponding elliptic curves may be recovered from some polynomial relationship between f and j: if the modular curve $X_0(N)$ has genus 0, then j is sometimes given by a rational formula in f; otherwise it has been suggested in [19] to look for a root of the modular polynomial $\Phi(f, j)$ after reducing modulo p and specialising in the value found for f in $\mathbb{F}_q$. (Both cases require that all coefficients be rational to make sense modulo p, which holds for all exhibited class invariants.)

2.3 Class polynomial computation over $\mathbb{C}$

This article is concerned only with the complexity of the class polynomial computation, for which a number of different algorithms have been suggested. The subsequent factorisation of the reduced polynomial in $\mathbb{F}_q$ and the construction of the corresponding CM curves use standard techniques and are not discussed any further.

The approach using complex floating point numbers is inspired directly by the facts of the previous sections. It consists of enumerating and normalising the τ_i representing the class group of $\mathcal{O}_D$, computing approximations to the $f(\tau_i) \in \mathbb{C}$ and to the class polynomial $H_D[f] = \prod_{i=1}^{h}(X - f(\tau_i)) \in \mathbb{C}[X]$ and rounding its coefficients to integers, either in $\mathbb{Z}$ or the maximal order of $\mathbb{Q}(\sqrt{D})$.

3 Basic complexity results

This section discusses the complexity of the basic multiprecision and polynomial arithmetic underlying the computations. The overall complexity of class polynomial computation is dominated by the approximation of the values of modular functions, discussed in Section 5. Throughout the remainder of this article we use llog as an abbreviation for $\log\log$, and lllog for $\log\log\log$.

3.1 Multiprecision floating point arithmetic

Let $M(n)$ be the bit complexity of multiplying two n-bit integers; Schönhage–Strassen multiplication implies $M(n) \in O(n \log n \operatorname{llog} n)$. Then the four basic arithmetic operations and the square roots of real floating point numbers of precision n have a complexity of $O(M(n))$, the inversion and square roots being realised by Newton iterations as explained in Lemmata 2.2 and 2.3 of [4]. The same article shows that exp, sin and the constant π can be computed in $O(\log n\, M(n))$.

Thus, the four basic operations on complex floating point numbers of precision n can be executed in time $O(M(n))$. Letting $c = \sqrt{\frac{a+\sqrt{a^2+b^2}}{2}}$ for $a > 0$, one obtains $\sqrt{a+bi} = c + \frac{b}{2c}$ in time $O(M(n))$. Finally, the complex exponential may be reduced to the real exponential and the real sine and cosine functions and is thus computed with complexity $O(\log n\, M(n))$.

3.2 Polynomial arithmetic

Multiplying two polynomials of degree d over $\mathbb{C}$ by the FFT takes $O(d \log d)$ multiplications in $\mathbb{C}$, whence it has a complexity of $O(d \log d\, M(n))$ once the necessary roots of unity have been computed.

The following algorithm obtains a monic polynomial of degree h from its roots by organising the computations in a binary tree.

Algorithm 1 (`Poly_from_roots`)
Input: *$h \in \mathbb{N}$ and $x_1, \ldots, x_h \in \mathbb{C}$*
Output: *binary tree $T_{k,i} = \big(X - x_{(i-1)2^k+1}\big) \cdots \big(X - x_{\min(i\,2^k,h)}\big)$*
for $k = 0, \ldots, t = \lceil \log_2 h \rceil$, $i = 1, \ldots, 2^{t-k}$;
in particular, the root $T_{t,1} = (X - x_1) \cdots (X - x_h)$

1. ***for*** $i = 1$ ***to*** h
   $T_{0,i} \leftarrow X - x_i$
2. ***for*** $i = h+1$ ***to*** 2^t
   $T_{0,i} \leftarrow 1$
3. ***for*** $k = 1$ ***to*** t
   for $i = 1$ ***to*** 2^{t-k}
   $T_{k,i} \leftarrow T_{k-1,2i-1} \cdot T_{k-1,2i}$
4. ***return*** T

If the needed roots of unity are precomputed, the algorithm has a complexity of $O(h \log^2 h\, M(n))$. Notice that the roots of unity of order $2^{\lceil \log_2 h \rceil}$ suffice to

carry out all the FFTs in Algorithm 1. By Section 3.1, a primitive root of unity takes $O(\log n\, M(n))$, all others can be obtained by successive multiplications in $O(h\, M(n))$. Thus, the total complexity of Algorithm 1 becomes

$$O((h \log^2 h + \log n) M(n)).$$

One of the asymptotically optimal algorithms for class polynomials relies on a related technique of symbolic computation (see [22, §10.1]). Let $f(X)$ be a polynomial of degree d over $\mathbb{C}$ that is to be evaluated in h different arguments $x_1, \ldots, x_h$. The key observation for doing so fast is that $f(x_i)$ is nothing but $f(X) \bmod X - x_i$. So the first step of the algorithm is the construction of the binary tree $T_{k,i}$ as in Algorithm 1, containing products of more and more of the $X - x_i$. Then a matching tree $R_{k,i} = f \bmod T_{k,i}$ is computed in the converse order, from its root to its leaves, such that the leaves contain the desired values of f.

Algorithm 2 (`Multi_eval`)
Input: $h \in \mathbb{N}$, $f \in \mathbb{C}[X]$ *and* $x_1, \ldots, x_h \in \mathbb{C}$
Output: $f(x_1), \ldots, f(x_h)$

1. $T \leftarrow$ `Poly_from_roots` $(h, x_1, \ldots, x_h)$
2. $t \leftarrow \lceil \log_2 h \rceil$, $R_{t,1} \leftarrow f \bmod T_{t,1}$
3. ***for*** $k = t - 1$ ***downto*** 0
   for $i = 1$ ***to*** 2^{t-k}
   $R_{k,i} \leftarrow R_{k+1, \lfloor \frac{i+1}{2} \rfloor} \bmod T_{k,i}$
4. ***return*** $R_{0,1}, \ldots, R_{0,h}$

A division with remainder of a polynomial of degree d by a polynomial of degree d' has the same complexity as multiplying polynomials of degree bounded by d ([22, §9.1]). The algorithm first computes an inverse modulo some power of X by Newton iterations and obtains the quotient with one multiplication; the remainder then requires another multiplication. So after the first reduction of f modulo the product of all the $X - x_i$, the complexity of Algorithm 2 on n-bit numbers is the same as that of Algorithm 1. Including the first reduction, it is given by

$$O((d \log d + h \log^2 h + \log n) M(n)).$$

4 Class group enumeration

Computing the class group of $\mathcal{O}_D$ by enumerating all reduced primitive quadratic forms of discriminant D is a step of the algorithm that can be neglected in practice. In the new algorithms of Sections 5.3 and 5.4, however, it risks to come close to the complexity of the crucial parts. One has to determine all coprime $[A_i, B_i, C_i]$ such that $|B_i| \le A_i \le C_i$, $B_i > 0$ if one of the inequalities is not strict, and $D = B_i^2 - 4A_iC_i$. This implies $A_i \le \sqrt{\frac{|D|}{3}}$. Looping over A and B such that $0 \le B \le A \le \sqrt{\frac{|D|}{3}}$ and computing the corresponding C, if it

exists, takes $O(|D|)$ arithmetic operations with integers of bit size in $O(\log|D|)$. The total complexity for this step becomes

$$O(|D|\, M(\log|D|)) = O(|D| \log|D| \operatorname{llog}|D| \operatorname{lllog}|D|),$$

which is smaller than the bound of Theorem 1 only by logarithmic factors.

To obtain an asymptotically faster algorithm, one loops over B alone and completely factors $\frac{B^2-D}{4}$. Using Strassen's deterministic algorithm of [37], one factorisation requires $O(|D|^{1/4} \log^k |D|)$ arithmetic operations for some k. One then constructs all divisors and considers them as candidates for A. Denote by $d(n)$ the number of divisors of an integer n; then $\frac{\log d(n)}{\log n} \to 0$ for $n \to \infty$, and $d(n) \in O(n^\varepsilon)$ for any $\varepsilon > 0$. The complexity of this algorithm is thus

$$O(|D|^{3/4} \log^k |D|\, M(\log|D|)) \subseteq O(|D|^{3/4+\varepsilon})$$

or, by Siegel's theorem,

$$O(h^{3/2+\varepsilon}) \text{ for any } \varepsilon > 0.$$

If one is willing to renounce at determinism, the subexponential randomised factoring algorithms of [33, 30] yield a complexity of $O(|D|^{1/2+\varepsilon})$ or $O(h^{1+\varepsilon})$ for the class group computation. This proves Theorem 3.

When working with a class invariant other than j that is not invariant under $\mathrm{Sl}_2(\mathbb{Z})$, but only under $\Gamma^0(N)$ for some $N > 0$, these representatives of the class group must be normalised, for instance by transforming them into an N-system following [35]. Such a system is defined by the property that

$$\gcd(A_i, N) = 1 \text{ and } B_i \equiv B_1 (\mathrm{mod} 2N) \text{ for } 1 \le i \le h$$

and may be obtained by applying suitable unimodular transformations to the original $[A_i, B_i, C_i]$. The coefficients of these transformations are defined modulo N, so that for fixed N, trying all possibilities requires a constant number of arithmetic operations per form and increases the size of the A_i, B_i and C_i by a constant factor. Thus, the total complexity of this step becomes $O(h\, M(\log|D|))$, which is negligible compared to the enumeration step. (In practice, one would use a more intelligent approach to lower the complexity with respect to N, cf. the constructive proof of Proposition 3 in [35].)

5 Complexity of class polynomial computation

Collecting the results of Sections 3 and 4, we are now able to provide the generic complexity for class polynomial computation via floating point approximations. Different approaches to evaluating the class invariants lead to algorithms with different overall complexities; these are examined below.

Let n be the precision in bits used for the computations; since we assume that rounding errors do not play a role, this is essentially the logarithmic height of the resulting class polynomial, see Section 6. Let h be the class number of $\mathcal{O}_D$ and denote by $E(h, n)$ the time needed to evaluate the class invariant with precision n in h values. Then the algorithm takes

- $O(h^{3/2+\varepsilon})$ for enumerating the class group according to Section 4;

- $E(h,n)$ for evaluating the class invariant and
- $O((h\log^2 h+\log n)M(n))$ for reconstructing the class polynomial from its roots according to Algorithm 1.

The required precision n is actually of the order of h (see Section 6), so that the class group computation indeed becomes negligible since $E(h,n)=\Omega(hn)=\Omega(h^2)$. The discussion of the following sections shows that the evaluations of the modular function also dominate the construction of the polynomial from its roots, so from now on we examine in more detail the quantity $E(h,n)$.

We propose four different algorithms for evaluating modular functions. The first two of them are well-known, the third one is a novel application of the techniques of symbolic computation presented in Section 3.2 and almost allows to obtain the complexity stated in Theorem 1. The fourth one finally proves Theorem 1 by gaining an additional logarithmic factor.

The class polynomial $H_D[f]=\prod_{i=1}^{h}(X-f(\tau_i))$ is obtained by evaluating the function f in the h different arguments $\tau_i=\frac{-B_i+\sqrt{D}}{2A_i}$ for an N-system $[A_i,B_i,C_i]$ (see Section 4), where N depends on f and is assumed to be fixed. Since $f(z)$ is modular for $\Gamma^0(N)$, it is invariant under the translation $z\mapsto z+N$ and admits a Fourier transform, that is, a Laurent series expansion in the variable $q^{1/N}=e^{2\pi iz/N}$. Thus, the $f(\tau_i)$ may be obtained by first computing the corresponding $q_i^{1/N}$ and then evaluating the q-expansion in these arguments. By Section 5.3, the q_i can be computed in time $O(h\log n\,M(n))=O(hn\log^2 n\,\mathrm{llog}\,n)$, which will be dominated by the actual function evaluations.

5.1 The naïve approach

The straightforward technique for evaluating $f=\sum_{\nu=\nu_0}^{\nu_1}c_\nu\left(q^{1/N}\right)^\nu$ consists of a Horner scheme for the polynomial part $\sum_{\nu=0}^{\nu_1-\nu_0}c_{\nu+\nu_0}\left(q^{1/N}\right)^\nu$ and a multiplication by $\left(q^{1/n}\right)^{\nu_0}$. (Notice that usually $\nu_0<0$.) Its complexity is $O((\nu_1-\nu_0+\log\nu_0)M(n))$, the c_ν having been precomputed to precision n. Actually, ν_0 is a constant depending only on f; ν_1, however, depends not only on the c_ν, but also on the desired precision n and on $|q^{1/N}|$ and thus on the function argument.

Consider first the classical case of f being j, which is invariant under $\mathrm{Sl}_2(\mathbb{Z})$. We may then assume that the arguments are transformed by a matrix in $\mathrm{Sl}_2(\mathbb{Z})$ into the standard fundamental domain

$$\mathcal{F}=\left\{z\in\mathbb{C}:|z|>=1,-\frac{1}{2}\le\Re(z)<\frac{1}{2}\right\}$$

prior to evaluating j. Then $|q|\le\left|e^{2\pi i\frac{1+\sqrt{-3}}{2}}\right|=e^{-\pi\sqrt{3}}=\text{const}$. On the other hand, it is shown in [6] that $0\le c_\nu\le\frac{e^{4\pi\sqrt{\nu}}}{\sqrt{2}\,\nu^{3/4}}$ for $\nu\ge1$. Thus, it is possible to fix $\nu_1\in O(n)$ to obtain a precision of $O(n)$ digits.

The total complexity of evaluating j at h values then becomes

$$O(hn\,M(n)),$$

or

$$O\left(|D|^{3/2}\log^6|D|\log\log|D|\right)$$

with $n = O(\sqrt{|D|}\log^2|D|)$ and $h = O(\sqrt{|D|}\log|D|)$ according to Section 6.

Concerning alternative class invariants, it holds unfortunately that the fundamental domain for $\Gamma^0(N)$ with $N > 1$ contains at least one rational number (called *cusp*). In such a cusp, $|q| = 1$, and the q-expansion usually diverges; in a neighbourhood of the cusp, it may converge arbitrarily slowly. In this case, it is possible to use a different expansion in the neighbourhood by transporting the cusp to infinity via a matrix in $\mathrm{Sl}_2(\mathbb{Z})$. We will not pursue this discussion, since the approach of Section 5.2 provides a faster and simpler solution for all class invariants actually suggested in the literature.

5.2 Using the sparsity of η

Virtually all class invariants suggested in the literature are in some way derived from Dedekind's η-function. This is the case for the Weber functions $\mathfrak{f}(z) = e^{-\pi i/24}\,\frac{\eta\left(\frac{z+1}{2}\right)}{\eta(z)}$, $\mathfrak{f}_1(z) = \frac{\eta\left(\frac{z}{2}\right)}{\eta(z)}$ and $\mathfrak{f}_2(z) = \sqrt{2}\,\frac{\eta(2z)}{\eta(z)}$ already examined in [38], the generalised Weber functions $\mathfrak{w}_N(z) = \frac{\eta\left(\frac{z}{N}\right)}{\eta(z)}$ suggested in [18], the double η quotients $\mathfrak{w}_{p_1,p_2}(z) = \frac{\eta\left(\frac{z}{p_1}\right)\eta\left(\frac{z}{p_2}\right)}{\eta(z)\eta\left(\frac{z}{p_1p_2}\right)}$ proposed in [19], and even for j. In fact, j is most conveniently computed as $j = \left(\frac{\mathfrak{f}_1^{24}+16}{\mathfrak{f}_1^8}\right)^3$.

The definition of η in [12] is closely related to the partition generating function:

$$\eta = q^{1/24}\prod_{\nu\geq 0}(1-q^\nu);$$

evaluating the product to precision n requires $O(n)$ arithmetic operations. An expression better suited for computation is given by Euler's pentagonal number theorem [21]:

$$\eta = q^{1/24}\left(1+\sum_{\nu=1}^{\infty}(-1)^\nu\left(q^{\nu(3\nu-1)/2}+q^{\nu(3\nu+1)/2}\right)\right).$$

Since the occurring exponents are values of quadratic polynomials, the series is very sparse: to reach an exponent of order $O(n)$, only $O(\sqrt{n})$ terms need to be computed, and this process can be implemented with $O(\sqrt{n})$ multiplications. Recall that any polynomial of fixed degree can be evaluated in an arithmetic progression with a constant number of arithmetic operations per additional value, once the first few values are known; the employed algorithm relies on iterated differences. In the special case of η, the following recursion yields two additional terms of the series at the expense of four multiplications by recursively computing q^ν, $q^{2\nu-1}$, $q^{\nu(3\nu-1)/2}$ and $q^{\nu(3\nu+1)/2}$ as follows:

$$\begin{aligned}
q^{\nu+1} &= q^\nu\cdot q\\
q^{2(\nu+1)-1} &= q^{2\nu-1}\cdot q^2\\
q^{\nu(3(\nu+1)-1)/2} &= q^{\nu(3\nu+1)/2}\cdot q^{2(\nu+1)-1}\\
q^{\nu(3(\nu+1)+1)/2} &= q^{\nu(3(\nu+1)-1)/2}\cdot q^{\nu+1}
\end{aligned}$$

Besides the sparse and regular series expression, the η function has a second crucial property that makes it well suited for computation: it is a modular form

of weight 1/2. As such, unlike j, it is not invariant under transformations in $\mathrm{Sl}_2(\mathbb{Z})$. However, its transformation behaviour is explicitly known (cf. [14, §4]) and easily computable. Thus, to obtain $\eta(z)$ for an arbitrary value of z, one should first transform z into the fundamental domain $\mathcal{F}$, so that the series can be truncated at an exponent of order $O(n)$. Then the evaluation of $O(\sqrt{n})$ terms of the η series for $O(h)$ distinct values (the constant being at most 4 for the $\mathfrak{w}_{p_1,p_2}$ mentioned in the beginning of this section) with a floating point precision of $O(n)$ digits can be carried out in

$$O(h\sqrt{n}\, M(n)),$$

or

$$O\left(|D|^{5/4} \log^5 |D| \log\log |D|\right)$$

as $h = O(\sqrt{|D|} \log |D|)$ and $n = O(\sqrt{|D|} \log^2 |D|)$ suffices according to Section 6.

It remains, however, to verify that transforming the arguments into the fundamental domain is dominated by the cost of the series evaluation. The arguments being roots of an N-system $[A_i, B_i, C_i]$ with $A_i, |B_i| \in O(N^2\sqrt{|D|}) = O(\sqrt{|D|})$, they may be transformed into $\mathcal{F}$ by symbolically reducing the quadratic forms in time $O(h\, M(\log |D|))$ (see [9, Prop. 5.4.3]), which is negligible. The same holds for arguments such as $\frac{z+1}{2}$ or $\frac{z}{N}$, corresponding to quadratic forms whose discriminants have absolute values in $O(|D|)$.

5.3 Multipoint evaluation

The algorithms of Sections 5.1 and 5.2 compute the values of modular functions one at a time; but for the sake of class polynomial computation, we need the values in many points. For polynomials, Algorithm 2 provides a fast way of doing exactly this. But from a numerical point of view, a class invariant can be seen as a polynomial via its truncated q-expansion. Either one considers the function directly as done for j in Section 5.1, or one proceeds via η as in Section 5.2. In both cases, a polynomial of degree $O(n)$ has to be evaluated in $O(h)$ points, which by Algorithm 2 can be done in time

$$\begin{aligned} & O((n \log n + h \log^2 h) M(n)) = O(n^2 \log^2 n \operatorname{llog} n + nh \log^2 h \log n \operatorname{llog} n) \\ = \; & O\left(|D| \log^6 |D| \log\log |D|\right). \end{aligned}$$

5.4 Newton iterations on the arithmetic-geometric mean

A new approach for evaluating modular functions in single arguments is described in [15]. It is based on the arithmetic-geometric mean and Newton iterations on a function involving it. The basic algorithm underlying Theorem 4 of [15] computes the modular function k', whose square λ satisfies

$$\frac{256\left(1-\lambda+\lambda^2\right)^3}{(\lambda(1-\lambda))^2} = j.$$

For an argument with imaginary part bounded by a constant and the precision n tending to infinity, it has a complexity of $O(\log n M(n))$.

During class polynomial computations, the precision and the imaginary part of the arguments are tightly coupled, so that this algorithm is not sufficient to derive the desired complexity result. The modification of Theorem 5 of [15] obtains the same complexity of $O(n \log^2 n \log\log n)$ uniformly in the argument. If the imaginary part of the argument is of the order of the required precision, then the algorithm of Section 5.2 or even the naïve algorithm of Section 5.1 already yield the desired result with a constant number of arithmetic operations. Otherwise, the argument is repeatedly divided by 2 until its imaginary part is smaller than a constant, which can be compensated by iterations of the arithmetic-geometric mean. Then the basic algorithm converges sufficiently fast.

For other modular functions f, one may have the evaluation of k' followed by Newton iterations on the modular polynomial relating k' and f. For fixed f, this phase does not increase the complexity. The approach does not work, however, for η, which is a modular form of weight $1/2$ instead of a modular function (of weight 0). In Section 7 of [15], it is suggested to obtain θ_{00}^2 as the inverse of the arithmetic-geometric mean of 1 and k', and then η as the twelfth root of $\lambda(1-\lambda)\theta_{00}^2/16$ by a suitably initialised Newton process. Again, one obtains a complexity of $O(\log n M(n))$ for an evaluation at precision n, uniformly in the argument.

The complexity for evaluating at h arguments then becomes

$$O(h \log n M(n)) = O\left(|D| \log^5 |D| \log\log |D|\right)$$

with the estimates of Section 6 for h and n. This proves Theorem 1.

6 The height of class polynomials

To complete the complexity analysis of the previous sections, we need to discuss the precision n at which the computations need to be carried out. Assuming that rounding errors can be neglected, the precision corresponds to the size of the largest coefficient of the class polynomial, or otherwise said, the logarithmic height of the polynomial (since its coefficients lie in $\mathbb{Z}$ or the maximal order of an imaginary-quadratic number field). It is sufficient to consider only the case of j. Indeed, as shown in [17], the height of class polynomials for a different f changes asymptotically by a constant factor, depending on the degrees in f and j of the modular polynomial connecting the two functions. This is crucial in practice, since the factor may reach 72, but may be neglected for the theoretical analysis.

Since the values of j are usually large, more often than not the largest coefficient of the class polynomial is its constant one. Then approximating $j(\tau) = q^{-1} + 744 + \sum_{\nu=1}^{\infty} c_\nu q^\nu$ by the first term, q^{-1}, one obtains a heuristic estimate for the height of

$$\pi\sqrt{|D|} \sum_{[A,B,C]} \frac{1}{A},$$

where the sum is taken over all reduced primitive quadratic forms of discriminant D. This estimate is very well confirmed by experimental findings, see [17].

The remainder of this section is devoted to derive a rigorous upper bound for the height of the class polynomials for j that matches as closely as possible the experimental truth.

The upper bound $\frac{e^{4\pi\sqrt{\nu}}}{\sqrt{2}\nu^{3/4}}$ on c_ν of [6] yields

$$|j(\tau) - q^{-1}| \leq 744 + \sum_{\nu=1}^{\infty} \frac{e^{4\pi\sqrt{\nu}}}{\sqrt{2}\nu^{3/4}} e^{-\pi\sqrt{3}} = c = 1370.566\ldots$$

for τ in the fundamental domain $\mathcal{F}$, so that

$$|j(\tau)| \leq |q^{-1}| + c \leq c'|q^{-1}|$$

with $c' = \frac{c}{e^{\pi\sqrt{3}}} = 5.939\ldots$

So the logarithmic height of the coefficient in front of X^k is bounded above by

$$\begin{aligned}
&\log\left(\binom{h}{k}\prod_{i=1}^{k}(c'|q_i^{-1}|)\right), \text{ assuming that the } A_i \text{ are in increasing order} \\
&\leq \log\left(2^h (c')^h \prod_{i=1}^{h} |q_i^{-1}|\right) \leq \log(2c')h + \pi\sqrt{|D|}\sum_{i=1}^{h}\frac{1}{A_i} \\
&= O\left(h + \sqrt{|D|}\sum_{i=1}^{h}\frac{1}{A_i}\right)
\end{aligned}$$

(As shown later, the first term of this bound is in fact dominated by the second one.)

The next step consists of estimating $\sum_{i=1}^{h}\frac{1}{A_i}$. To this purpose, consider the number of possible B's for a given A. This number is certainly bounded above by the number of $B \in]-A, A]$ that satisfy $B = D \bmod 2$ and $\frac{B^2-D}{4} = 0 \bmod A$. Assuming the worst case that the quadratic equation has a root modulo each prime divisor of A and considering the cases of odd and even A separately, one obtains an upper bound on the number of B's of $2 \cdot 2^{\omega(A)} \leq 2\tau(A)$, where $\omega(A)$ denotes the number of prime factors of A and $\tau(A)$ its number of divisors. Hence,

$$\sum_{i=1}^{h}\frac{1}{A_i} \leq 2\sum_{A=1}^{\sqrt{\frac{|D|}{3}}}\frac{\tau(A)}{A},$$

and this sum may be bounded by standard techniques from analytic number theory. We use the estimates $\sum_{n=1}^{N}\frac{1}{n} = \log N + \gamma + O(1/N)$ with $\gamma = 0.577\ldots$ the Euler-Mascheroni constant and $\sum_{n=1}^{N}\frac{\log n}{n} = \frac{1}{2}\log^2 N + O(1)$. Then,

$$\begin{aligned}
\sum_{A=1}^{N}\frac{\tau(A)}{A} &= \sum_{A=1}^{N}\sum_{1\leq m,n:mn=A}\frac{1}{mn} = \sum_{1\leq m,n:mn\leq N}\frac{1}{mn} = \sum_{m=1}^{N}\frac{1}{m}\sum_{n=1}^{\lfloor N/m\rfloor}\frac{1}{n} \\
&= \frac{1}{2}\log^2 N + 2\gamma\log N + O(1) \\
&= O(\log^2 N).
\end{aligned}$$

We have thus shown that the height is bounded by

$$\log(2c')h + \pi\sqrt{|D|}\log^2|D| + O\left(\sqrt{|D|}\log|D|\right)$$

Similarly to the argumentation above one shows $\sum_{A=1}^{n} \tau(A) = O(N \log N)$ and thus $h = O(\sqrt{|D|} \log |D|)$. Hence a bound on the height is given by

$$O\left(\sqrt{|D|} \log^2 |D|\right).$$

This proves Theorem 2.

Bounding the number of B's for a given A by $2 \cdot 2^{\omega(A)}$ is much beyond what can be expected on average. It corresponds to the case where the equation has a solution modulo all prime divisors of A. Taking into account that this case occurs with probability about $\frac{1}{2^{\omega(A)}}$ and that in all other cases there is no solution, one would rather expect a constant number of B's per A. Then $\sum_{i=1}^{h} \frac{1}{A_i}$ behaves as $\sum_{n=1}^{\sqrt{\frac{|D|}{3}}} \frac{1}{n} \in O(\log |D|)$. So on average, one would expect to gain a logarithmic factor with respect to the worst case estimation, which would result in a complexity of

$$O(|D| \log^4 |D| \log \log |D|)$$

in Theorem 1.

7 Implementation

The algorithms of this article have been implemented using `gmp` [27] with an assembly patch for 64 bit AMD processors [24], `mpfr` [28] and `mpc` [20] for the multiprecision arithmetic and a library of the author's for the polynomial operations. Table 1 provides running times for class numbers between 2500 and 100000, obtained on an AMD Opteron 250 with 2.4 GHz. All timings are given in seconds and rounded to two significant digits. (The computations for class number 100000 have been carried out on a 2.2 GHz machine disposing of more memory, and the running times have been scaled accordingly.) For each class number, the discriminant with smallest absolute value has been chosen. Only the algorithms of Sections 5.2 to 5.4 are taken into account; the naïve approach of Section 5.1 is clearly inferior to the one exploiting the sparsity of η. The chosen class invariant is the double η-quotient $\mathfrak{w}_{3,13}$, and its values are obtained by precomputing a table for the values of η at the h reduced quadratic forms.

The first lines of the table provide some general information. The precision (1) of the floating point computations is obtained by increasing the estimate of Section 6 by 1% to account for potential rounding errors. As the chosen class invariant is not j, a correction factor depending on the class invariant needs to be used, see Section 6. This factor is correct only asymptotically, which explains why the actual height (2) is a bit smaller than the precision estimate. For j, the two are closer to each other. Lines (3) and (4) give information useful for the later discussion. $M(n)$ is measured by computing the first 100000 successive powers of $\pi + i\gamma$ with the Euler-Mascheroni constant γ.

The second block of lines provides timings for the steps that are independent of the algorithm used for evaluating the modular function. As can be seen, the class group computation (5) is completely negligible. Line (6) corresponds to the effort of deriving all values of the class invariant from the tabulated η values (reduction of quadratic forms, multiplication by 24-th roots of unity and computation of the η quotients). The computation of the polynomial from its

		5000	10000	20000	40000	100000
	$\lvert D\rvert$	6961631	23512271	98016239	357116231	2093236031
(1)	precision n (bits)	9540	20317	45179	96701	264727
(2)	height	8431	18114	40764	87842	242410
(3)	$n/\sqrt{\lvert D\rvert}$	3.6	4.2	4.6	5.1	5.8
(4)	$M(n)$	7.3	23	75	230	1080
(5)	class group	0.1	0.1	0.4	1.3	6.8
(6)	conjugates from η	3.4	21	140	890	10000
(7)	poly. from roots	13	93	730	5200	120000
	sparse series					
(8)	η	12	98	900	7700	140000
(9)	of which q_i	3.0	22	170	1300	20000
(10)	total time	28	210	1800	14000	270000
	multipoint eval.					
(11)	η	93	640	5700	42000	aborted
(12)	total time	110	750	6500	48000	
	(11) / (8)	7.8	6.5	6.3	5.5	—
	AGM					
(13)	η	32	200	1400	9900	130000
(14)	total time	48	320	2300	16000	260000
	(13) / (8)	2.7	2.0	1.6	1.3	0.93

Table 1: Running times

roots (7) corresponds precisely to Algorithm 1, and provides a measure for the complexity of the operations with polynomials. For constructing the largest polynomial of degree 100000 from its roots, the polynomial FFT has been disabled during the last steps and replaced by Toom–Cook multiplication, since the FFT consumed too much memory; this explains the jump from $h = 40000$.

The third block contains the timings for evaluating η in the reduced quadratic forms (8) using the sparse series representation as described in Section 5.2, and the total running time for computing the class polynomials using this technique (10). Line (9) details the time spent in (8) (and also in (11)) for computing the q_i; it essentially measures the complex exponential.

The fourth block represents the corresponding results for the multipoint evaluation approach of Section 5.3, and the last block corresponds to the asymptotically optimal evaluation of Section 5.4, for which an implementation by Dupont has been used. As explained in Section 5.4, the algorithm requires to switch to the sparse series evaluation when the imaginary part of the argument becomes too large. In the implementation, the AGM code is disabled for an imaginary part larger than 5.

Comparing first the evaluation of η as a sparse series or by multipoint evaluation, one notices that the asymptotically faster algorithm is about 5 to 8 times slower on the examples and that it appears to catch up with growing class numbers. However, this happens so slowly that one cannot expect it to beat the algorithm in $O(|D|^{1.25+\varepsilon})$ in the foreseeable future for any tractable instances.

The approach using Newton iterations on the AGM is faster than multipoint evaluation, but still hardly beats the asymptotically slower algorithm in $O\left(|D|^{1.25+\varepsilon}\right)$: the biggest computed example of class number 100000 lies just beyond the cross-over point!

One notices that the growth rates of the running times of all algorithms, instead of behaving like $|D|$ or $|D|^{1.25}$, come closer to $|D|^{1.4}$ or $|D|^{1.6}$. A small part of this can be explained by the rather peculiar choice of discriminants. Taking the first one with a given class number, the precision n is rather large compared to h and $\sqrt{|D|}$, since there are many forms with small A so that the sum $\sum_{i=1}^{h} \frac{1}{A_i}$ becomes comparatively large. Apparently, this effect becomes more pronounced for growing class numbers; the ratio $n/\sqrt{|D|}$ grows from 3.6 to 5.8 in the examples.

However, the major reason for the faster than predicted growth of computing time is that for the floating point arithmetic the range for asymptotically fast algorithms is not yet reached. The threshold for switching to the FFT in `gmp` on the test machine is set to about 500000 bits; so the examples lie still in the Karatsuba or Toom-Cook range, which accounts for a growth of $M(n)$ of $n^{\log 3/\log 2}$ resp. $n^{\log 5/\log 3}$ instead of n.

A possible improvement of the multipoint evaluation approach consists of conveniently grouping the arguments. For instance, in the example of class number 5000, the function η has to be evaluated in 2501 arguments (corresponding to the two ambiguous forms and 2499 pairs of opposite non-ambiguous forms). Assuming the worst case of $|q| \approx e^{-\pi\sqrt{3}}$, that is almost reached for the largest values of A, one can approximate η by a polynomial of degree 1190. As multipoint evaluation should be most efficient when the number of arguments is about half the degree, it makes sense to perform four evaluations in 625 resp. 626 arguments each. But sorting them by their absolute values, the smaller ones do not actually require such a high degree approximation of η. In the example, an approximation of degree 260, 551, 805 resp. 1190 is sufficient for the four chunks of q. Then the time used for multipoint evaluation drops from 93 s to 64 s. Experimenting with different partitions of the arguments (three resp. five parts of the same size, parts of different sizes adapted to the degree of the approximations, etc.) yields similar results, far from competing with the sparse series evaluation.

Another point to take into account are the space requirements of the algorithms. When each root of the class polynomial is computed separately, only $O(hn)$ bits need to be stored, which is linear in the output size. Multipoint evaluation as described in Section 3.2, however, requires that the tree constructed in Step 1 of Algorithm 2 be maintained in memory, so that the occupied space grows by a logarithmic factor to reach $O(hn \log h)$. This logarithmic factor could be saved by evaluating in chunks of $O\left(\frac{h}{\log h}\right)$ arguments, as explained in Lemma 2.1 of [23].

Anyway, the example class polynomial of degree 100000 uses over 5 GB as an uncompressed text file and is computed in about 3 days. This shows that the limiting factor is the memory requirement rather than the running time, as can be expected from algorithms that have a close to linear complexity with respect to their output size.

As a final remark, one notices that the algorithms behave numerically well, even though rounding errors do occur during floating point computations. For the algorithm of Section 5.2, this can be explained by the sparsity of the η series and the fact that all coefficients are $+1$ and -1. Indeed, if the last few digits of a term are erroneous, these errors propagate to subsequent terms. However, the

absolute magnitude of such errors decreases rapidly, so that the wrong digits in later terms actually do not intervene in the additions. (Otherwise said, the computations may as well be carried out with fixed point numbers, and indeed a simulation of fixed point arithmetic using floating point numbers of decreasing precision yields accurate results.)

8 Comparison to other approaches

8.1 Chinese remaindering

In [1], the authors suggest an approach for directly computing class polynomials for j modulo a prime p. The basic idea is to derive the polynomial modulo many small primes by enumerating all elliptic curves modulo these small primes and only retaining those having complex multiplication by $\mathcal{O}_D$. Then a Chinese remainder technique allows to obtain the class polynomial modulo p. The first step unfortunately does not depend on p: so many small primes are needed that in principle the class polynomial could be reconstructed over $\mathbb{Z}$. The complexity derived in Section 3.2 of [1] is

$$O\left(|D|^{3/2}\log^{10}|D| + |D|\log^2|D|\log p + \sqrt{|D|}\log^2 p\right),$$

and already the term depending only on $|D|$ is worse than what is obtained with the practically employed algorithm of Section 5.2, not to speak of the asymptotically fast versions of Sections 5.3 and 5.4.

Thus, it is asymptotically faster to compute the class polynomial over $\mathbb{Z}$ using floating point approximations and to reduce it modulo p afterwards.

8.2 p-adic algorithms

Couveignes and Henocq in [10] suggest a p-adic approach for computing class polynomials for j. The basic idea is to look for a small prime p and (by exhaustive search) an elliptic curve modulo p with complex multiplication by $\mathcal{O}_D$. This curve is then lifted to the p-adic numbers with a high enough precision so that the class polynomial may be reconstructed. Alternatively, they show how to work with supersingular curves. The complexity of their approach is $O(|D|^{1+\varepsilon})$, where the exact power of the logarithmic factor has not been worked out. So the complex and the p-adic approach are both essentially linear in the size of the class polynomials. All variants have been implemented (for more details on the complex implementation, see [17]; for the ordinary p-adic algorithm, see [7]; for the supersingular one, [31]), and all seem to work quite well in practice. The main advantage of the complex approach for the time being is that it can easily be extended to work with arbitrary class invariants using the results of [35]. First steps towards using alternative class invariants in the ordinary p-adic algorithm are undertaken in [7], but they remain to be fully automated.

Acknowledgements

I am grateful to Régis Dupont for letting me use his code to test the fast evaluation of η via the arithmetic-geometric mean. I thank Karim Belabas for

pointing out to me the basic idea of Section 6 and Pierrick Gaudry and François Morain for numerous discussions.

References

[1] Amod Agashe, Kristin Lauter, and Ramarathnam Venkatesan. Constructing elliptic curves with a known number of points over a prime field. In Alf van der Poorten and Andreas Stein, editors,*High Primes and Misdemeanours: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams*, volume 41 of*Fields Institute Communications*. American Mathematical Society, 2004.

[2] A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Mathematics of Computation*, 61(203):29–68, July 1993.

[3] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks — Third International Conference, SCN 2002, Amalfi, Italy, September 2002*, volume 2576 of*Lecture Notes in Computer Science*, pages 257–267, Berlin, 2003. Springer-Verlag.

[4] Richard P. Brent. Fast multiple-precision evaluation of elementary functions. *Journal of the ACM*, 23(2):242–251, 1976.

[5] Friederike Brezing and Annegret Weng. Elliptic curves suitable for pairing based cryptography. *Designs, Codes and Cryptography*, 37(1):133–141, 2005.

[6] N. Brisebarre and G. Philibert. Effective lower and upper bounds for the-Fourier coefficients of powers of the modular invariant j. *Journal of the Ramanujan Mathematical Society*, 20:255–282, 2005.

[7] Reinier Bröker and Peter Stevenhagen. Elliptic curves with a given number of points. In Duncan Buell, editor,*Algorithmic Number Theory — ANTS-VI*, volume 3076 of*Lecture Notes in Computer Science*, pages 117–131, Berlin, 2004. Springer-Verlag.

[8] Jinhui Chao, Osamu Nakamura, Kohji Sobataka, and Shigeo Tsujii. Construction of secure elliptic cryptosystems usingCM tests and liftings. In Kazuo Ohta and Dingyi Pei, editors,*Advances in Cryptology — ASIACRYPT '98*, volume 1514 of*Lecture Notes in Computer Science*, pages 95–109, Berlin, 1998. Springer-Verlag.

[9] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics. Springer-Verlag, New York, 1993.

[10] Jean-Marc Couveignes and Thierry Henocq. Action of modular correspondences aroundCM points. In Claus Fieker and David R. Kohel, editors,*Algorithmic Number Theory — ANTS-V*, volume 2369 of*Lecture Notes in Computer Science*, pages 234–243, Berlin, 2002. Springer-Verlag.

[11] David A. Cox. *Primes of the Form x^2+ny^2 —Fermat, Class Field Theory, and Complex Multiplication.* John Wiley & Sons, New York, 1989.

[12] R. Dedekind. Erläuterungen zu den vorstehendenFragmenten. In R. Dedekind and H. Weber, editors,*Bernhard Riemann's gesammelte mathematische Werke und wissenschaftlicher Nachlaß*, pages 438–447. Teubner, Leipzig, 1876.

[13] Max Deuring. DieTypenderMultiplikatorenringeelliptischer Funktionenkörper. *Abhandlungen aus dem mathematischen Seminar der hamburgischen Universität*, 14:197–272, 1941.

[14] Max Deuring. DieKlassenkörper der komplexenMultiplikation. In*Enzyklop. d. math. Wissenschaften*, volume I 2 Heft 10. Teubner, Stuttgart, 2 edition, 1958.

[15] Régis Dupont. Fast evaluation of modular functions usingNewton iterations and theAGM. To appear in *Mathematics of Computation*; available at `http://www.lix.polytechnique.fr/Labo/Regis.Dupont/preprints/Dupont_FastEvalMod.ps.gz`, 2005.

[16] Régis Dupont, Andreas Enge, and François Morain. Building curves with arbitrary smallMOV degree over finite prime fields. *Journal of Cryptology*, 18(2):79–89, 2005.

[17] Andreas Enge and François Morain. Comparing invariants for class fields of imaginary quadratic fields. In Claus Fieker and David R. Kohel, editors,*Algorithmic Number Theory — ANTS-V*, volume 2369 of*Lecture Notes in Computer Science*, pages 252–266, Berlin, 2002. Springer-Verlag.

[18] Andreas Enge and François Morain. Further investigations of the generalisedWeber functions. In preparation, 2005.

[19] Andreas Enge and Reinhard Schertz. Constructing elliptic curves over finite fields using double eta-quotients. *Journal de Théorie des Nombres de Bordeaux*, 16:555–568, 2004.

[20] Andreas Enge and Paul Zimmermann. `mpc` — a library for multiprecision complex arithmetic with exact rounding. Version 0.4.5, available at `http://www.lix.polytechnique.fr/Labo/Andreas.Enge/Software.html`.

[21] Leonhard Euler. Evolutio producti infiniti $(1-x)(1-xx)(1-x^3)(1-x^4)(1-x^5)(1-x^6)$ etc. in seriem simplicem. *Acta academiae scientiarum Petropolitanae*, 1780:I:125–169, 1783. *Opera Omnia* I.3:472–479.

[22] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, 1999.

[23] Joachim von zur Gathen and Victor Shoup. ComputingFrobenius maps and factoring polynomials. *Computational Complexity*, 2:187–224, 1992.

[24] Pierrick Gaudry. Assembly support forGMP onAMD64, April 2005. Available at `http://www.lix.polytechnique.fr/Labo/Pierrick.Gaudry/mpn_AMD64/`.

[25] Alice Gee. Class invariants byShimura's reciprocity law. *Journal de Théorie des Nombres de Bordeaux*, 11(1):45–72, 1999.

[26] Alice Gee and Peter Stevenhagen. Generating class fields usingShimura reciprocity. In J. P. Buhler, editor,*Algorithmic Number Theory — ANTS-III*, volume 1423 of*Lecture Notes in Computer Science*, pages 441–453, Berlin, 1998. Springer-Verlag.

[27] TorbjörnGranlund et. al. `gmp` —GNU multiprecision library. Version 4.1.4, available at `http://www.swox.com/gmp`.

[28] Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and PaulZimmermann et. al. `mpfr` — a library for multiple-precision floating-point computations with exact rounding. Version 2.2.0, available at `http://www.mpfr.org`.

[29] A. K. Lenstra and H. W.Lenstra Jr. Algorithms in number theory. In Jan van Leeuwen, editor,*Algorithms and Complexity*, volume A of*Handbook of Theoretical Computer Science*, chapter 12, pages 673–715. Elsevier, Amsterdam, 1990.

[30] H. W.Lenstra Jr. and Carl Pomerance. A rigorous time bound for factoring integers. *Journal of the American Mathematical Society*, 5(3):483–516, 1992.

[31] R. Lercier and E. Riboulet-Deyris. Elliptic curves with complex multiplication. Posting to the Number Theory List, available at `http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0401&L=nmbrthry&P=R305`, 2004.

[32] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces forFR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–1243, May 2001.

[33] Carl Pomerance. Fast, rigorous factorization and discrete logarithm algorithms. In David S. Johnson, Takao Nishizeki, Akihiro Nozaki, and Herbert S. Wolf, editors,*Discrete Algorithms and Complexity, Proceedings of the Japan–US Joint Seminar, June 4–6, 1986, Kyoto, Japan*, volume 15 of*Perspectives in Computing*, pages 119–143, Orlando, 1987. Academic Press.

[34] Reinhard Schertz. Die singulärenWerte derWeberschenFunktionen $\mathfrak{f}$, $\mathfrak{f}_1$, $\mathfrak{f}_2$, γ_2, γ_3. *Journal für die reine und angewandte Mathematik*, 286/287:46–74, 1976.

[35] Reinhard Schertz. Weber's class invariants revisited. *Journal de Théorie des Nombres de Bordeaux*, 14(1):325–343, 2002.

[36] Carl Ludwig Siegel. Über dieClassenzahl quadratischerZahlkörper. *Acta Arithmetica*, 1:83–86, 1936.

[37] Volker Strassen. EinigeResultate überBerechnungskomplexität. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 78:1–8, 1976.

[38] Heinrich Weber. *Lehrbuch der Algebra*, volume 3. Chelsea Publishing Company, New York, 3rd edition, 1908.