FAST EVALUATION OF MODULAR FUNCTIONS USING NEWTON ITERATIONS AND THE AGM

RÉGIS DUPONT

Abstract. We present an asymptotically fast algorithm for the numerical evaluation of modular functions such as the elliptic modular function j. Our algorithm makes use of the natural connection between the arithmetic-geometric mean (AGM) of complex numbers and modular functions. Through a detailed complexity analysis, we prove that for a given τ, evaluating N significative bits of $j(\tau)$ can be done in time $O(\mathcal{M}(N)\log N)$, where $\mathcal{M}(N)$ is the time complexity for the multiplication of two N-bit integers. However, this is only true for a *fixed* τ and the time complexity of this first algorithm greatly increases as $\mathrm{Im}(\tau)$ does. We then describe a second algorithm that achieves the same time complexity independently of the value of τ in the classical fundamental domain $\mathcal{F}$. We also show how our method can be used to evaluate other modular forms, such as the Dedekind η function, with the same time complexity.

1. Introduction

In this article, we study the complexity of the numerical evaluation of modular functions at complex values: given an element τ lying in $\mathcal{F} = \{z \in \mathbb{C} : \mathrm{Im}(z) > 0, |z| \geq 1, |\mathrm{Re}(z)| \leq 1/2\}$ and an integer $N \geq 0$, we investigate the time complexity for the evaluation of N significative bits of modulars functions such as j at τ. Our main result is a constructive proof of the following theorem:

Theorem. *Given an integer $N \geq 0$ and an element $\tau \in \mathcal{F}$, one can compute $J \in \mathbb{C}$ such that*

$$\left|\frac{j(\tau) - J}{j(\tau)}\right| \leq \frac{1}{2^N}$$

in time $O(\mathcal{M}(N)\log N)$, where $\mathcal{M}(N)$ is the time complexity for the multiplication of two N-bit integers[*].

This work was motivated in part by applications such as the computation of Hilbert class polynomials [7] and the computation of modular polynomials [8], where it can be necessary to evaluate modular functions to precisions in the tens of thousands of bits.

We begin by recalling a few facts about modular functions and theta constants, before giving a first (naïve) algorithm for the evaluation of a class of modular functions, with a running time in $O(\mathcal{M}(N)\sqrt{N})$ where N is the required precision. We then study some properties of the arithmetic-geometric mean (AGM), show how it can be used to evaluate a particular modular function using Newton iterations and give a first algorithm, which time complexity is thoroughly analyzed. This first algorithm achieves the announced complexity in $O(\mathcal{M}(N)\log N)$ but only in the case where τ is fixed. We then propose a second algorithm having the same running time, independently of the value of $\tau \in \mathcal{F}$. Finally, we show how this method can be used to evaluate other modular functions or forms, such as the Dedekind η function, and we give some experimental results.

An important feature of our algorithm is that it can be generalized to genus 2 (see the forthcoming [6]).

2. Modular functions and theta constants

2.1. Modular functions.

In this section, we briefly recall a few facts about modular functions and fix some notation. We refer the reader to [15] for a complete introduction to the theory of modular functions.

Throughout this paper, we let $\mathcal{H}$ denote the Poincaré half-plane: $\mathcal{H} = \{z \in \mathbb{C} : \mathrm{Im}(z) > 0\}$. For $\tau \in \mathcal{H}$, we let $q = q(\tau) = \exp(i\pi\tau)$ (note that this differs from modern conventions, where an extra factor of 2 is

2000 *Mathematics Subject Classification.* Primary 65D20; Secondary 33E05, 11Y16.

Key words and phrases. Modular functions, numerical evaluation, AGM, Newton iterations.

[*]Typically —using FFT-based multiplication—, we have $\mathcal{M}(N) = O(N\log N\log\log N) = O(N^{1+\epsilon})$.

added). We denote by Γ the full elliptic modular group, that is the quotient of $\mathrm{SL}_2(\mathbb{Z})$ by $\langle -I \rangle$, and from now on we identify a matrix in $\mathrm{SL}_2(\mathbb{Z})$ with its class in Γ. The action of Γ on $\mathcal{H}$ is given by:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \tau = \frac{a\tau + b}{c\tau + d},$$

for every $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma$ and $\tau \in \mathcal{H}$.

If we introduce the region $\mathcal{F} \subset \mathcal{H}$:

$$\mathcal{F} = \{\tau \in \mathcal{H} : |\mathrm{Re}(\tau)| \leq 1/2, |\tau| \geq 1\},$$

then $\mathcal{F} \setminus (\partial\mathcal{F} \cap \{\tau \in \mathcal{H} : \mathrm{Re}(\tau) > 0\})$ is a fundamental domain for the action of Γ on $\mathcal{H}$.

We also introduce the generators S and T of Γ, defined by:

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

and

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

so that for every $\tau \in \mathcal{H}$, $T\tau = \tau + 1$ and $S\tau = -1/\tau$.

If Γ' is a subgroup of finite index of Γ, then a function $f : \mathcal{H} \to \mathbb{C}$ is said to be modular for Γ' if it is meromorphic on $\mathcal{H}$, invariant under the action induced by Γ' (*i.e.*, for all $\gamma \in \Gamma'$ and $\tau \in \mathcal{H}$, $f(\gamma\tau) = f(\tau)$), and meromorphic at the cusps.

2.2. **Theta constants with characteristic 2.** The so-called *theta constants with characteristic 2* are the three functions defined, for $\tau \in \mathcal{H}$ and $q = \exp(i\pi\tau)$, by:

$$\theta_{00}(\tau) = \sum_{n\in\mathbb{Z}} q^{n^2},$$

$$\theta_{01}(\tau) = \sum_{n\in\mathbb{Z}} (-1)^n q^{n^2},$$

and

$$\theta_{10}(\tau) = \sum_{n\in\mathbb{Z}} q^{(n+\frac{1}{2})^2}.$$

For an extensive treatment of these functions, we refer the reader to [14] or [16], which contain the proofs of all the few properties we now state (we only give proofs when elementary ones exists, that we are aware of).

Proposition 1. *The functions θ_{00}, θ_{01} and θ_{10} are analytic on $\mathcal{H}$ and do not vanish.*

Proof. That these functions are analytic on $\mathcal{H}$ follows from their definition as q-series. That they do not vanish on $\mathcal{H}$ follows from [14], Lemma 4.1. □

Proposition 2. *For all $\tau \in \mathcal{H}$,*

$$\theta_{00}^2(S\tau) = -i\tau\theta_{00}^2(\tau),$$
$$\theta_{01}^2(S\tau) = -i\tau\theta_{10}^2(\tau),$$
$$\theta_{10}^2(S\tau) = -i\tau\theta_{01}^2(\tau).$$

Proof. See [16, pp. 103–105]. □

Proposition 3. *For all $\tau \in \mathcal{H}$,*

$$\theta_{00}^4(\tau) = \theta_{01}^4(\tau) + \theta_{10}^4(\tau).$$

Proof. See [16, pp. 71–76]. □

Proposition 4.

$$\lim_{\mathrm{Im}(\tau)\to+\infty} \theta_{00}(\tau) = 1,$$
$$\lim_{\mathrm{Im}(\tau)\to+\infty} \theta_{01}(\tau) = 1,$$

and

$$\lim_{\mathrm{Im}(\tau)\to+\infty} \theta_{10}(\tau) = 0.$$

Proof. This is easily proven from the definitions of the theta functions as q-series. □

Proposition 5. *For all $\tau \in \mathcal{H}$,*

$$\theta_{00}^2(2\tau) = \frac{\theta_{00}^2(\tau) + \theta_{01}^2(\tau)}{2}$$

and

$$\theta_{01}^2(2\tau) = \theta_{00}(\tau)\theta_{01}(\tau).$$

Proof. This is easily derived from the definitions of θ_{00} and θ_{01}, using the identity $2(n^2+m^2) = (n+m)^2 + (n-m)^2$. □

Proposition 6. *For all $\tau \in \mathcal{H}$,*

$$4\frac{d}{d\tau}\log\frac{\theta_{10}}{\theta_{01}}(\tau) = i\pi\theta_{00}^4(\tau),$$
$$4\frac{d}{d\tau}\log\frac{\theta_{00}}{\theta_{01}}(\tau) = i\pi\theta_{10}^4(\tau),$$
$$4\frac{d}{d\tau}\log\frac{\theta_{10}}{\theta_{00}}(\tau) = i\pi\theta_{01}^4(\tau).$$

Proof. See [16, p. 82]. □

2.3. **The functions k and k'.** Jacobi's functions k and k' are defined, for $\tau \in \mathcal{H}$, by:

$$k(\tau) = \left(\frac{\theta_{10}(\tau)}{\theta_{00}(\tau)}\right)^2 \tag{1}$$

and

$$k'(\tau) = \left(\frac{\theta_{01}(\tau)}{\theta_{00}(\tau)}\right)^2. \tag{2}$$

From Propositions 2 and 3, it follows directly that, for all $\tau \in \mathcal{H}$,

- $k^2(\tau) + k'^2(\tau) = 1$,
- $k(S\tau) = k'(\tau)$ and $k'(S\tau) = k(\tau)$.

Moreover, k' and k are modular functions:

Proposition 7. *The functions k' and k are modular for the groups $\Gamma_{k'}$ and Γ_k, where*

$$\Gamma_{k'} = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma : a \equiv d \equiv 1 \bmod 4, b \equiv 0 \bmod 2, c \equiv 0 \bmod 4 \right\}$$

and

$$\Gamma_{k} = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma : a \equiv d \equiv 1 \bmod 4, b \equiv 0 \bmod 4, c \equiv 0 \bmod 2 \right\}.$$

Moreover, if

$$\mathcal{F}' = \{\tau \in \mathcal{H} : |\mathrm{Re}(z)| < 1, |2\tau+1| > 1, |2\tau-1| > 1\},$$

then

$$k'(\mathcal{F}') = k(\mathcal{F}') = \{z \in \mathbb{C} \setminus \{0,1\} : \mathrm{Re}(z) > 0\},$$

and the real part of $k(\tau)$ (resp. of $k'(\tau)$) vanishes on the boundary of $\mathcal{F}'$.

Proof. The results for k' are proved in [4], Lemmata 2.7 and 2.8. The results for k follow easily, using $k(\tau) = k'(S\tau)$ and the identities $\Gamma_k = S\Gamma_{k'}S$ and $S\mathcal{F}' = \mathcal{F}'$. □

Finally, the well-known *elliptic modular function* j (modular for the full group Γ) can be defined by:

$$j(\tau) = 256\frac{\left(1 - k'^2(\tau) + k'^4(\tau)\right)^3}{k'^4(\tau)\left(1 - k'^2(\tau)\right)^2}. \tag{3}$$

(see [2, pages 112–116] for a proof of the fact that j is indeed modular for Γ).

3. Basic evaluation of j using the Dedekind η-function

3.1. **Notions of precision.** Our goal in this paper is to evaluate some functions with a given precision. We now define precisely what we mean by precision:

- We will say that $\alpha \in \mathbb{C}$ is an approximation of $a \in \mathbb{C}$ with *absolute* precision N bits if
$$|a - \alpha| \leq \frac{1}{2^N}.$$
- We will say that $\alpha \in \mathbb{C}$ is an approximation of $a \in \mathbb{C}$ with *relative* precision N bits if
$$\left|\frac{a - \alpha}{a}\right| \leq \frac{1}{2^N}.$$

We will make use of the following facts, whose proofs are straightforward:

- If α and β are approximations of a and $b \in \mathbb{C}$ with relative precision N bits ($N \geq 2$), then $\alpha\beta$ is an approximation of ab with relative precision $N - 2$ bits.
- If $n \geq 1$ and α is an approximation of $a \in \mathbb{C}$ with relative precision N bits ($N \geq n$), then α^n is an approximation of a^n with relative precision $N - n$ bits.
- If $\alpha \neq 0$ is an approximation of $a \in \mathbb{C} \setminus \{0\}$ with relative precision N bits ($N \geq 1$), then α^{-1} is an approximation of a^{-1} with relative precision $N - 1$ bits.
- If α is an approximation of $a \in \mathbb{C}$ with relative precision N bits, and if $\sqrt{a}$ is any of the two square roots of a, then the square root of α lying in the same quarter-plane as $\sqrt{a}$ is an approximation of $\sqrt{a}$ with relative precision N bits.
- If α and β are approximations of a and $b \in \mathbb{C}$ with absolute precision N bits ($N \geq 1$), then $\alpha + \beta$ is an approximation of $a + b$ with absolute precision $N - 1$ bits.

One of the most delicate issues when studying algorithms to numerically evaluate functions is to know to what precision each computation has to be carried out. Due to what precedes, we know that precision losses will occur, hence the working precision will usually have to be higher than the precision required for the result. However, since the time cost of arithmetic operations increases with the working precision, one should always work at the minimal possible precision ensuring that the final result will be correct to the required precision.

In what follows, the precision at which arithmetic operations have to be carried out is not specified in the algorithms we give, but is discussed after the description of these algoritms.

3.2. **A naïve algorithm.** It is well known that the function j has a q-series expansion with integral coefficients:

$$j(\tau) = \frac{1}{q^2} + 744 + 169884q^2 + \ldots,$$

the coefficients of this series being effectively computable (see for example [1]).

However, this leads to an algorithm that is quite inefficient, since the series we consider is dense and the height of its coefficients grows quite fast: the time complexity would be in $O(\mathcal{M}(N)N) = O(N^{2+\varepsilon})$, where N is the required precision in bits.

A better approach is to use the functions θ_{00} and θ_{01}: their q-series can indeed be evaluated much faster than the one for j, since they are sparse with non-nil coefficients ± 1. The function k' can be computed as the square of their quotient, and j can then be evaluated using Equation (3).

For $\varepsilon \in \{0, 1\}$ and $B \geq 1$, we define the partial sums

$$S_{\varepsilon,B}(\tau) = \begin{cases} 1 & \text{if} \quad B = 1 \\ 1 + 2\sum_{n=1}^{B-1}(-1)^{\varepsilon n} q^{n^2} & \text{if} \quad B \geq 2 \end{cases}$$

for $\tau \in \mathcal{F}$.

Notice that if $\tau \in \mathcal{F}$, then $\mathrm{Im}(\tau) \geq \sqrt{3}/2$, hence $|q| \leq Q = \exp(-\pi\sqrt{3}/2)$. Thus, using the inequality

$$\left|\sum_{n \geq B} q^{n^2}\right| \leq \frac{|q|^{B^2}}{1-|q|} \leq 2\,|q|^{B^2}$$

we obtain:

$$|\theta_{0\varepsilon}(\tau) - S_{\varepsilon,B}(\tau)| \leq 4\,|q|^{B^2}.$$

Notice that in the case $B = 1$, this implies

$$|\theta_{0\varepsilon}(\tau) - 1| \leq 4\,|q| \leq 4Q \leq 0.27,$$

hence $\theta_{0\varepsilon}(\tau)$ is "close" to 1: for all $\tau \in \mathcal{F}$, any approximation of $\theta_{0\varepsilon}$ with absolute precision $N \geq 1$ bits is also an approximation with relative precision $N-1$ bits.

Now a straightforward computation shows that for all $\tau \in \mathcal{F}$, $N \in \mathbb{N}$ and $\varepsilon \in \{0,1\}$, if

$$B \geq \sqrt{\frac{N+3}{\pi \mathrm{Im}(\tau)\log_2 e}}$$

then

$$\left|\frac{\theta_{0\varepsilon}(\tau) - S_{\varepsilon,B}(\tau)}{\theta_{0\varepsilon}(\tau)}\right| \leq \frac{1}{2^N}$$

for $\varepsilon \in \{0,1\}$.

This shows that Algorithm 1 can be used to evaluate θ_{00} and θ_{01}.

Algorithm: `EvaluateThetaNaive`
input : $\tau \in \mathcal{F}$, $N \in \mathbb{N}$
output: (θ_0, θ_1) such that $|\theta_{0\varepsilon}(\tau) - \theta_\varepsilon| \,/\, |\theta_{0\varepsilon}(\tau)| \leq 2^{-N}$ for $\varepsilon \in \{0,1\}$

$q \leftarrow \exp(i\pi\tau)$;
$q_2 \leftarrow q^2$;
$q_a \leftarrow 1$;
$\theta_0 \leftarrow q$;
$\theta_1 \leftarrow -q$;
$B \leftarrow \lceil \sqrt{(N+3)/(\pi\mathrm{Im}(\tau)\log_2 e)} \rceil$;
for $n = 2$ **to** M **do**
  $q_a \leftarrow q_a q_2$;
  $q \leftarrow q_a q$;
  $\theta_0 \leftarrow \theta_0 + q$;
  $\theta_1 \leftarrow \theta_1 + (-1)^n q$;
end
$\theta_0 \leftarrow 1 + 2\theta_0$;
$\theta_1 \leftarrow 1 + 2\theta_1$;
return (θ_0, θ_1);

Algorithm 1: naïve evaluation of θ_{00} and θ_{01}

We now discuss the precision at which the computations have to be carried out. From what we have seen before, the final result should be correct with absolute precision at least $N+1$ bits. Clearly, this will be achieved if all computations are carried out at fixed precision $N + 5\log_2 M = O(N)$ bits. Since the constant π as well as the exponential function can be evaluated with relative precision N bits in time $O(\mathcal{M}(N)\log N)$, as explained in [3] for example, the time complexity of algorithm `EvaluateThetaNaive` (Algorithm 1) is in $O(\mathcal{M}(N)\sqrt{N})$.

We now turn to the evaluation of k': since $k'(\tau) = \left(\frac{\theta_{01}(\tau)}{\theta_{00}(\tau)}\right)^2$, and considering the precision losses that can occur, Algorithm 2 can be used to evaluate k':

Algorithm: `EvaluatekpNaive`
input : $\tau \in \mathcal{F}$, $N \in \mathbb{N}$
output: kp such that $|k'(\tau) - kp| / |k'(\tau)| \leq 2^{-N}$

$(\theta_0, \theta_1) \leftarrow$ `EvaluateThetaNaive`$(\tau, N+5)$;
$kp \leftarrow (\theta_1/\theta_0)^2$;
return kp;

Algorithm 2: naïve evaluation of k'

Clearly, algorithm `EvaluatekpNaive` (Algorithm 2) has time complexity in $O(\mathcal{M}(N)\sqrt{N})$.

Now j can be evaluated using Equation (3). However, one must be careful about the precision loss that can occur: as $\mathrm{Im}(\tau)$ increases, $\theta_{00}(\tau)$ and $\theta_{01}(\tau)$ will tend to 1, hence $1 - k'(\tau)^2$ will tend to zero. More precisely, we have $1 - k'^2(\tau) \sim 16q$ as $\mathrm{Im}(\tau) \to +\infty$, hence this operation can entail a big precision loss. In fact, in order to evaluate $j(\tau)$ with relative precision N bits, it is sufficient to have an evaluation of $k'(\tau)$ with relative precision $N + 5\mathrm{Im}(\tau)$ bits, and the algorithm is of course straightforward.

Remark 1. The j function can also be defined by means of the Dedekind η function, which shares a few properties with theta constants (sparse q series development,...). It then follows that using the η function q-series instead of the theta constants'ones yields another algorithm for the evaluation of j and related functions. This algorithm is described in [7] and has the same asymptotic complexity (up to a constant) as `EvaluatekpNaive`.

Remark 2. At first sight, the complexity of this method for the evaluation of j depends on the value of $\mathrm{Im}(\tau)$. Note, however, that the number B of iterations in Algorithm 1 is in $\sqrt{N/\mathrm{Im}(\tau)}$. In particular, if we fix some constant $K > 0$ and we want to evaluate the theta constants at $\tau \in \mathcal{F}$ with relative precision $N \leq K\mathrm{Im}(\tau)$ bits, then the number of iterations will be bounded by a constant. Using this remark, it is easy to see that the complexity of the evaluation of j is in $O(\mathrm{M}(N)\sqrt{N})$.

4. The arithmetic-geometric mean

4.1. **Definition.** Let a and b be two positive real numbers. The *arithmetic-geometric mean* (AGM) of a and b, uncovered by Lagrange, then rediscovered and much studied by Gauss, is the common limit of the two sequences (a_n) and (b_n) defined by $a_0 = a$, $b_0 = b$,

$$a_{n+1} = \frac{a_n + b_n}{2}$$

and

$$b_{n+1} = \sqrt{a_n b_n}.$$

A proof of the fact that these sequences indeed converge to a common limit can be found in [11, III, pp. 361–363] or in [2, 4].

The AGM can be extended to the case where a and b are complex numbers, although such a generalization is not straightforward (this is mainly due to the fact that, for every $n \in \mathbb{N}$, b_{n+1} is defined as being one of the two possible square roots of $a_n b_n$, hence the sequences (a_n) and (b_n) are not uniquely determined by a and b). A study of the AGM for complex numbers can again be found in [11, III, pp. 361–480] or in [4].

In order to simplify things somewhat, we only consider the case where a and b lie in $\mathcal{R} = \{z \in \mathbb{C} : \mathrm{Re}(z) > 0\}$. For $z \in \mathbb{C}$, we denote by $\sqrt{z}$ the unique square root of z lying in $\mathcal{R} \cup \mathbb{R}^{\geq 0}$.

We then define the sequences (a_n) and (b_n) as above. We have the following result:

Theorem 1. *Let $a, b \in \mathcal{R}$, and define the sequences (a_n) and (b_n) as above. Then they converge to a common limit* $\mathrm{AGM}(a, b) \in \mathcal{R}$*, and this convergence is quadratic: for all $n \in \mathbb{N}$,*

$$|a_{n+1} - b_{n+1}| \leq \frac{\pi\,|a_n - b_n|^2}{8\,\mathrm{Min}(|a|, |b|)}.$$

Proof. For every $n \in \mathbb{N}$, let $\theta_n \in [0, \pi[$ denote the unoriented angle between a_n and b_n. Proposition 2.1 of [4] proves that (a_n) and (b_n) converge to a common limit $\mathrm{AGM}(a,b) \in \mathcal{R}$, and the proof also shows that:

$$|a_{n+1} - b_{n+1}| \leq \frac{|a_n - b_n|}{2}$$

and

$$\theta_{n+1} \leq \frac{\theta_n}{2}.$$

For $n \in \mathbb{N}$, we have:

$$\begin{aligned} |a_{n+1} - b_{n+1}| &= \frac{\left|a_n + b_n - 2\sqrt{a_n b_n}\right|}{2} \\ &= \frac{\left|\sqrt{a_n} - \sqrt{b_n}\right|^2}{2} \\ &= \frac{|a_n - b_n|^2}{2\left|\sqrt{a_n} + \sqrt{b_n}\right|^2} \end{aligned}$$

Now, the unoriented angle between $\sqrt{a_n}$ and $\sqrt{b_n}$ is equal to $\theta_n/2 < \pi/2$, so that

$$\begin{aligned} \left|\sqrt{a_n} + \sqrt{b_n}\right|^2 &= |\sqrt{a_n}|^2 + \left|\sqrt{b_n}\right|^2 + 2\left|\sqrt{a_n b_n}\right| \cos(\theta_n/2) \\ &\geq |a_n| + |b_n| \\ &\geq 2\,\mathrm{Min}(|a_n|, |b_n|). \end{aligned}$$

From the proof of Proposition 2.1 of [4], we also have that

$$\mathrm{Min}\left(|a_n|, |b_n|\right) \geq \frac{\sin\theta_0}{\theta_0}\mathrm{Min}\left(|a|, |b|\right) \geq \frac{2}{\pi}\mathrm{Min}\left(|a|, |b|\right),$$

whence

$$\left|\sqrt{a_n} + \sqrt{b_n}\right|^2 \geq \frac{4}{\pi}\,\mathrm{Min}(|a|, |b|),$$

and finally:

$$|a_{n+1} - b_{n+1}| \leq \frac{\pi\,|a_n - b_n|^2}{8\,\mathrm{Min}(|a|, |b|)}.$$

□

The limit $\mathrm{AGM}(a,b)$ for a and $b \in \mathcal{R}$, is again called the arithmetic-geometric mean of a and b.

4.2. **Relationship with theta constants and other properties.** In order to simplify notations, we let $M : \mathcal{R} \to \mathcal{R}$ denote the (univariate) AGM† defined by $M(z) = \mathrm{AGM}(1, z)$.

The key point in the study of the AGM, relating it to modular functions and modular forms, is the following result:

Theorem 2. *For $\tau \in \mathcal{F}'$, we have:*

$$M\left(k'(\tau)\right) = \frac{1}{\theta_{00}^2(\tau)}.$$

Proof. Let $\tau \in \mathcal{H}$, and define the sequences (a_n) and (b_n) by

$$a_n = \frac{\theta_{00}^2(2^n\tau)}{\theta_{00}^2(\tau)}$$

and

$$b_n = \frac{\theta_{01}^2(2^n\tau)}{\theta_{00}^2(\tau)}.$$

†Considering a univariate AGM makes sense, since the bivariate AGM is homogenous: for all $a, b \in \mathbb{C}$, $\mathrm{AGM}(a,b) = a\mathrm{AGM}(1, b/a)$.

Proposition 4 shows that

$$\lim_{n\to\infty} \theta_{00}(2^n\tau) = \lim_{n\to\infty} \theta_{01}(2^n\tau) = 1,$$

whence

$$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = \frac{1}{\theta_{00}^2(\tau)}.$$

Since $k'(\tau) = \frac{\theta_{01}^2(\tau)}{\theta_{00}^2(\tau)}$, it is clear that $a_0 = 1$ and $b_0 = k'(\tau)$. Moreover, Proposition 5 proves that, for any $n \in \mathbb{N}$, we have:

$$a_{n+1} = \frac{a_n + b_n}{2}$$

and

$$b_{n+1}^2 = a_n b_n,$$

hence the key point in obtaining the announced result is proving that, if $\tau \in \mathcal{F}'$, then for any $n \in \mathbb{N}$, a_n and b_n lie in $\mathcal{R}$. This is contained in the proof of Theorem 2.2 of [4], more precisely in the proof of Lemma 2.9, that also makes use of Lemmata 2.3 to 2.8. □

Proposition 8. *The function M is analytic on $\mathcal{R} \setminus \{1\}$.*

Proof. We first prove that the Inverse Function Theorem can be applied to the function k' at any point in $\mathcal{F}'$. This is due to the fact that k' is analytic on $\mathcal{F}'$, and that, for all $\tau \in \mathcal{F}'$,

$$\frac{dk'}{d\tau}(\tau) = \frac{-i\pi\theta_{01}^2(\tau)\theta_{10}^4(\tau)}{2\theta_{00}^2(\tau)} \neq 0$$

(this expression of the derivative of k' is obtained using Proposition 6).

Now the function $1/\theta_{00}^2$ is also analytic on $\mathcal{F}'$, and for all $\tau \in \mathcal{F}'$, we have

$$M\left(k'(\tau)\right) = \frac{1}{\theta_{00}^2(\tau)},$$

hence the proposition follows from the fact that k' is surjective from $\mathcal{F}'$ to $\mathcal{R} \setminus \{1\}$. □

Proposition 9. *For $\tau \in \mathcal{F}'$, we have:*

$$\frac{dM}{dz}\left(k'(\tau)\right) = \frac{4\theta_{00}'(\tau)}{i\pi\theta_{00}(\tau)\theta_{01}^2(\tau)\theta_{10}^4(\tau)},$$

where $\theta_{00}'(\tau) = \frac{d\theta_{00}}{d\tau}(\tau)$.

Proof. This follows directly from the differentiation of the equality:

$$M\left(k'(\tau)\right) = \frac{1}{\theta_{00}^2(\tau)},$$

using the definition of the function k' and Proposition 6. □

5. Evaluating k' using Newton iterations and the AGM

In this section, we restrict to the case $\tau \in \mathcal{F}$. Using the same arguments as in Section 3, it is easy to see that for all $\tau \in \mathcal{F}$,

$$|k'(\tau) - 1| \leq 0.9,$$

hence any approximation of $k'(\tau)$ with absolute precision $N + 4$ bits is also an approximation with relative precision N bits.

5.1. **A function vanishing at $k'(\tau)$.** Let $\tau \in \mathcal{F}$, since $\mathcal{F} \subset \mathcal{F}'$ and $S\mathcal{F} \subset \mathcal{F}'$, we have $k'(\tau) \in \mathcal{R}$ and $k'(S\tau) \in \mathcal{R}$, hence using the results on k and k' from Section 2.3 we can write $k'(S\tau) = \sqrt{1 - k'^2(\tau)}$.

Theorem 2 shows that:

$$M\left(k'(\tau)\right) = \frac{1}{\theta_{00}^2(\tau)}$$

and

$$M\left(k'(S\tau)\right) = \frac{1}{\theta_{00}^2(S\tau)},$$

hence, using Proposition 2, we have:

$$\frac{M\left(k'(\tau)\right)}{M\left(k'(S\tau)\right)} = -i\tau.$$

We now introduce the function f_τ defined for $z \in \mathcal{R}$ by:

$$f_\tau(z) = iM(z) - \tau M\left(\sqrt{1 - z^2}\right).$$

From the preceding discussion, it is clear that f_τ vanishes at $k'(\tau)$, hence the idea to use Newton iterations on f_τ in order to evaluate k' at τ. In order to show that this is indeed possible, we now give a few results about f_τ.

Proposition 10. *The function f_τ is analytic on $\mathcal{R} \setminus \{1\}$.*

Proof. This is a direct consequence of the definition of f_τ and of Proposition 8. □

Proposition 11. *For all $\tau \in \mathcal{F}$,*

$$\frac{df_\tau}{dz}\left(k'(\tau)\right) = \frac{-2}{\pi\tau\theta_{01}^2(\tau)\theta_{10}^4(\tau)}.$$

Proof. Differentiating the definition of f_τ, we get:

$$\frac{df_\tau}{dz}(z) = i\frac{dM}{dz}(z) + \frac{\tau z}{\sqrt{1 - z^2}}\frac{dM}{dz}\left(\sqrt{1 - z^2}\right),$$

hence:

$$\frac{df_\tau}{dz}\left(k'(\tau)\right) = i\frac{dM}{dz}\left(k'(\tau)\right) + \frac{\tau k'(\tau)}{k'(S\tau)}\frac{dM}{dz}\left(k'(S\tau)\right).$$

Using the expression for $\frac{dM}{dz}(k'(\tau))$ obtained in Proposition 9, we get:

$$\frac{df_\tau}{dz}\left(k'(\tau)\right) = \frac{4\theta_{00}'(\tau)}{\pi\theta_{00}(\tau)\theta_{01}^2(\tau)\theta_{10}^4(\tau)} + \frac{4\tau k'(\tau)\theta_{00}'(S\tau)}{i\pi k'(S\tau)\theta_{00}(S\tau)\theta_{01}^2(S\tau)\theta_{10}^4(S\tau)}$$

Now, differentiating the identity

$$\theta_{00}^2(S\tau) = -i\tau\theta_{00}^2(\tau)$$

gives us:

$$2\theta_{00}(S\tau)\theta_{00}'(S\tau) = -i\tau^2\theta_{00}(\tau)\left(\theta_{00}(\tau) + 2\tau\theta_{00}'(\tau)\right).$$

Using the equalities

$$k'(\tau) = \frac{\theta_{01}^2(\tau)}{\theta_{00}^2(\tau)}$$

and

$$k'(S\tau) = k(\tau) = \frac{\theta_{10}^2(\tau)}{\theta_{00}^2(\tau)},$$

and the transformation formulae given in Proposition 2, we obtain the announced result. □

We now introduce the function $g : \mathcal{R} \setminus \{1\} \to \mathbb{C}$ defined by

$$g(z) = \frac{M(z)^3}{z(1 - z^2)}$$

for all $z \in \mathcal{R} \setminus \{1\}$.

It is easy to see that g is analytic on $\mathcal{R} \setminus \{1\}$. Now, using the definitions of k and k' and Theorem 2 we have, for all $\tau \in \mathcal{F}$,

$$g\left(k'(\tau)\right) = \frac{1}{\theta_{01}^2(\tau)\theta_{10}^4(\tau)},$$

hence from Proposition 11:

$$\frac{df_\tau}{dz}\left(k'(\tau)\right) = \frac{-2}{\pi\tau} g\left(k'(\tau)\right).$$

5.2. **Newton iterations for the function f_τ.** In this section, we fix $\tau \in \mathcal{F}$, and write $\xi = k'(\tau)$.

Since f_τ is analytic on $\mathcal{R}$ and vanishes at ξ, Newton iterations on f_τ are a method of choice for the evaluation of k' at τ. Classically, this is done by first setting z_0 to some approximation of ξ, then by considering iterations of the form

$$z_{n+1} = z_n - \frac{f_\tau(z_n)}{f'_\tau(z_n)}.$$

If z_0 is sufficiently close to ξ, then the sequence (z_n) will converge to ξ, and the convergence will be quadratic.

It can be proven that if we replace $f'_\tau(z_n)$ by $f'_\tau(\xi)$ in each iteration, then the same result holds. However, this is usually not very useful since ξ is precisely the quantity we want to compute. In our case, we consider iterations of the form

$$z_{n+1} = z_n + \frac{\pi\tau f_\tau(z_n)}{2g(z_n)}.$$

The idea is that, since z_n is an approximation of ξ, then $\frac{-2}{\pi\tau}g(z_n)$ is an approximation of $\frac{-2}{\pi\tau}g(\xi) = f'_\tau(\xi)$.

More precisely, if we introduce the values $F(\tau)$ and $G(\tau)$ defined for all $\tau \in \mathcal{F}$ by[‡]

$$F(\tau) = \operatorname*{Sup}_{n\geq 2} \left| \frac{f_\tau^{(n)}(\xi)}{n! f'_\tau(\xi)} \right|^{\frac{1}{n-1}}$$

and

$$G(\tau) = \operatorname*{Sup}_{n\geq 1} \left| \frac{g^{(n)}(\xi)}{n! g(\xi)} \right|^{\frac{1}{n}},$$

we have the following result:

Theorem 3. *Let $\tau \in \mathcal{F}$ and $\xi = k'(\tau)$. Define the sequence (z_n) by $z_0 \in \mathcal{R} \setminus \{1\}$ and*

$$z_{n+1} = z_n + \frac{\pi\tau f_\tau(z_n)}{2g(z_n)}$$

for all $n \in \mathbb{N}$. Then, if

$$|\xi - z_0| \leq \frac{1}{6\operatorname{Max}(F(\tau), G(\tau))}$$

and

$$|\xi - z_0| < \frac{1}{2}\operatorname{Min}\left(|\xi|, |1-\xi|\right),$$

the sequence (z_n) is well defined and we have, for all $n \geq 0$:

$$|\xi - z_n| \leq \left(\frac{1}{2}\right)^{2^n - 1} |\xi - z_0|.$$

In order to prove this theorem, we first prove the following lemma:

Lemma 1. *Let $\tau \in \mathcal{F}$, $\xi = k'(\tau)$ and $\alpha = \operatorname{Max}\left(F(\tau), G(\tau)\right)$. Then for all $z \in \mathcal{R} \setminus \{1\}$ such that*

$$2\alpha\,|\xi - z| < 1,$$

we have

$$\left|\xi - z + \frac{\pi\tau f_\tau(z)}{2g(z)}\right| \leq \frac{2\alpha\,|z-\xi|^2}{1 - 2\alpha\,|z-\xi|}.$$

[‡]The fact that $F(\tau)$ and $G(\tau)$ are indeed finite follows from the analycity of f_τ and g.

Proof. Let $\tau \in \mathcal{F}$, $\xi = k'(\tau)$ and $\alpha = \operatorname{Max}(F(\tau), G(\tau))$. We have

$$\begin{aligned}
\left|\xi - z + \frac{\pi\tau f_\tau(z)}{2g(z)}\right| &= \left|\frac{\pi\tau}{2g(z)}\right| \cdot \left|f_\tau(z) - (z-\xi)\left(\frac{-2}{\pi\tau}g(z)\right)\right| \\
&\leq \left|\frac{\pi\tau}{2g(z)}\right| \left(\left|f_\tau(z) - (z-\xi)\left(\frac{-2}{\pi\tau}g(\xi)\right)\right| + |z-\xi| \cdot \left|\frac{2}{\pi\tau}\right| \cdot |g(z) - g(\xi)|\right) \\
&\leq \left|\frac{g(\xi)}{g(z)}\right| \left(\frac{|f_\tau(z) - (z-\xi)f'_\tau(\xi)|}{|f'_\tau(\xi)|} + |x-\xi| \cdot \frac{|g(z)-g(\xi)|}{|g(\xi)|}\right).
\end{aligned}$$

Now, consider the Taylor expansion for f_τ at ξ:

$$f_\tau(z) = (z-\xi)f'_\tau(\xi) + \sum_{n\geq 2} \frac{f_\tau^{(n)}(\xi)}{n!}(z-\xi)^n,$$

hence

$$\begin{aligned}
\left|\frac{f_\tau(z) - (z-\xi)f'_\tau(\xi)}{f'_\tau(\xi)}\right| &\leq \sum_{n\geq 2} \left|\frac{f_\tau^{(n)}(\xi)}{n! f'_\tau(\xi)}\right| \cdot |z-\xi|^n \\
&\leq \sum_{n\geq 2} F(\tau)^{n-1} |z-\xi|^n \\
&\leq \frac{F(\tau)|z-\xi|^2}{1 - F(\tau)|z-\xi|},
\end{aligned}$$

since, by hypothesis, $F(\tau)|z-\xi| < 1$.

In a similar fashion, by considering the Taylor expansion for g at ξ and using the definition of $G(\tau)$, we get

$$\frac{g(z)}{g(\xi)} = 1 + \sum_{n\geq 1} \frac{g^{(n)}(\xi)}{n! g(\xi)}(z-\xi)^n,$$

where

$$\left|\sum_{n\geq 1} \frac{g^{(n)}(\xi)}{n! g(\xi)}(z-\xi)^n\right| \leq \frac{G(\tau)|z-\xi|}{1 - G(\tau)|z-\xi|}.$$

The hypothesis on $|z-\xi|$ implies that

$$\frac{G(\tau)|z-\xi|}{1 - G(\tau)|z-\xi|} < 1,$$

hence we get

$$\left|\frac{g(\xi)}{g(z)}\right| \leq \frac{1}{1 - \frac{G(\tau)|z-\xi|}{1-G(\tau)|z-\xi|}} = \frac{1 - G(\tau)|z-\xi|}{1 - 2G(\tau)|z-\xi|}.$$

These inequalities, combined with the fact that the functions $x \mapsto \frac{x}{1-x}$ and $x \mapsto \frac{1-x}{1-2x}$ are increasing on $[0, 1/2[$ and the definition of α, prove that

$$\left|\xi - z + \frac{\pi\tau f_\tau(z)}{2g(z)}\right| \leq \frac{2\alpha|z-\xi|^2}{1 - 2\alpha|z-\xi|}.$$

□

Proof [of Theorem 3]. Let $\tau \in \mathcal{F}$, $\xi = k'(\tau)$, $\alpha = \operatorname{Max}(F(\tau), G(\tau))$, and $z_0 \in \mathcal{R} \setminus \{1\}$ such that

- $6\alpha|\xi - z_0| < 1$,
- $|\xi - z_0| < |z_0|$,
- $|\xi - z_0| < |1 - z_0|$.

We define the sequence (z_n) by

$$z_{n+1} = z_n + \frac{\pi \tau f_\tau(z_n)}{2g(z_n)}$$

for all $n \in \mathbb{N}$, and let $\lambda = 2\alpha\,|\xi - z_0|$.

We now prove, by induction on n, that for all $n \in \mathbb{N}$, $z_n \in \mathcal{R} \setminus 1$ (*i.e.*, the sequence is well defined), and

$$|\xi - z_n| \leq \left(\frac{\lambda}{1-\lambda}\right)^{2^n - 1} |\xi - z_0|\,. \tag{4}$$

This is clearly true for $n = 0$, we now show that if it holds for n, then it holds for $n+1$.

Since $z_n \in \mathcal{R} \setminus \{1\}$, then z_{n+1} is well defined. Since $6\alpha\,|\xi - z_0| < 1$, we have $\frac{\lambda}{1-\lambda} < 1$ hence $|\xi - z_n| \leq |\xi - z_0|$, and Lemma 1 can be applied to z_n to show that

$$|\xi - z_{n+1}| \leq \frac{2\alpha\,|\xi - z_n|^2}{1 - 2\alpha\,|\xi - z_n|}.$$

Combining this with the fact that Equation (4) holds for z_n, we get

$$\begin{aligned}
|\xi - z_{n+1}| &\leq \frac{2\alpha}{1 - 2\alpha\,|\xi - z_n|}\left(\left(\frac{\lambda}{1-\lambda}\right)^{2^n-1}|\xi - z_0|\right)^2 \\
&\leq \frac{2\alpha}{1 - 2\alpha\,|\xi - z_n|}\left(\frac{\lambda}{1-\lambda}\right)^{2^{n+1}-2}|\xi - z_0|^2 \\
&\leq \left(\frac{\lambda}{1-\lambda}\right)^{2^{n+1}-1}|\xi - z_0|\,.
\end{aligned}$$

Now, the condition $6\alpha\,|\xi - z_0| < 1$ ensures that $\frac{\lambda}{1-\lambda} < \frac{1}{2}$, hence

$$|\xi - z_{n+1}| \leq \left(\frac{1}{2}\right)^{2^{n+1}-1}|\xi - z_0| \leq |\xi - z_0|\,.$$

Since $|\xi - z_0| < \frac{|\xi|}{2}$ and $|\xi - z_0| < \frac{|1-\xi|}{2}$, then z_{n+1} lies in $\mathcal{R} \setminus \{1\}$, and the theorem is proved. □

6. Algorithms and complexity results for the evaluation of k'

6.1. **Evaluation of the AGM.** In this section, we let $z \in \mathcal{R}$ and we denote by (a_n) and (b_n) the sequences associated with the computation of $M(z)$ (letting $a_0 = 1$ and $b_0 = z$).

We begin by giving an upper bound on the number of AGM iterations required in order to compute $M(z)$ to a given relative precision:

Proposition 12. *For all $z \in \mathcal{R}$, $N \in \mathbb{N}$, if we denote by (a_n) and (b_n) the sequences associated with the computation of $M(z)$ (with $a_0 = 1$ and $b_0 = z$) and define n_N by*

$$n_N = \mathrm{Max}\left(\lceil \log_2 |\log_2 |z|| \rceil, 1\right) + \lceil \log_2(N+3) - 1 \rceil,$$

then a_{n_N} is an approximation of $M(z)$ to relative precision N bits.

Proof. Let $z \in \mathcal{R}$ and $N \geq 0$. As usual, we denote by (a_n) and (b_n) the AGM sequences associated with the computation of $M(z)$, and we introduce the following auxilliary sequences:

- $(m_n)_{n\in\mathbb{N}}$ (resp. $(M_n)_{n\in\mathbb{N}}$), defined by $m_n = \mathrm{Min}\left(|a_n|, |b_n|\right)$ (resp. $M_n = \mathrm{Max}\left(|a_n|, |b_n|\right)$) for all $n \geq 0$,
- $(c_n)_{n\in\mathbb{N}}$ defined by $c_n = M_n/m_n$ for all $n \geq 0$, and
- $(\theta_n)_{n\in\mathbb{N}}$, where θ_n is the unoriented angle between a_n and b_n.

The sequence (M_n) is clearly decreasing while, from the proof of Proposition 2.1 of [4],

$$|a_{n+1} - b_{n+1}| \leq \frac{|a_n - b_n|}{2}, \tag{5}$$

$$m_{n+1} \geq m_n \cos \frac{\theta_n}{2}, \tag{6}$$

$$m_n \geq \frac{\sin \theta_0}{\theta_0} m_0 \geq \frac{2}{\pi} m_0 \tag{7}$$

and $\theta_{n+1} \leq \frac{\theta_n}{2}$ for all $n \geq 0$.

We then have, for $n \geq 0$,

$$\left| \frac{a_{n+1}}{b_{n+1}} \right|^2 = \frac{|a_n + b_n|^2}{4\,|a_n b_n|} \leq \frac{(2M_n)^2}{4M_n m_n} = c_n$$

and

$$\left| \frac{b_{n+1}}{a_{n+1}} \right|^2 = \frac{4\,|a_n b_n|}{|a_n + b_n|^2} \leq \frac{4M_n m_n}{4m_n^2 \cos^2 \theta_n} = \frac{c_n}{\cos^2 \theta_n},$$

where we have used the fact that

$$|a_n + b_n|^2 = |a_n|^2 + |b_n|^2 + 2\,|a_n b_n| \cos \theta_n \geq 2m_n^2(1 + \cos \theta_n) = 4m_n^2 \cos^2 \theta_n.$$

Using the fact that $\theta_n \leq \frac{\pi}{2}$, this proves that

$$c_{n+1} \leq \sqrt{2c_n}. \tag{8}$$

For all $n \geq 0$, we have

$$|a_{n+1} - b_{n+1}| = \frac{\left|\sqrt{a_n} - \sqrt{b_n}\right|^2}{2} = \frac{|a_n - b_n|^2}{2\left|\sqrt{a_n} + \sqrt{b_n}\right|^2},$$

and

$$\left|\sqrt{a_n} + \sqrt{b_n}\right|^2 = |a_n| + |b_n| + 2\left|\sqrt{a_n}\sqrt{b_n} \cos \frac{\theta_n}{2}\right| \geq 2m_n \left(1 + \cos \frac{\theta_n}{2}\right) = 4m_n \cos^2 \frac{\theta_n}{4},$$

so (using the fact that $\frac{\theta_n}{4} \leq \frac{\pi}{8}$),

$$|a_{n+1} - b_{n+1}| \leq \frac{|a_n - b_n|^2}{\alpha m_n}, \tag{9}$$

where $\alpha = 8 \cos^2 \frac{\pi}{8} = 4\left(1 + \frac{1}{\sqrt{2}}\right)$.

Now, let $n_0 = \mathrm{Max}\left(\lceil \log_2 |\log_2 |z|| \rceil, 1\right)$: since $\log_2 c_0 = |\log_2 |z||$, then (by an induction using Equation (8)) $\log_2 c_{n_0} \leq 1$ hence $c_{n_0} \leq 2$. Moreover,

$$|a_{n_0} - b_{n_0}|^2 = |a_{n_0}|^2 + |b_{n_0}|^2 - 2\,|a_{n_0} b_{n_0}| \cos \theta_{n_0} = M_{n_0}^2 + m_{n_0}^2 - 2M_{n_0} m_{n_0} \cos \theta_{n_0},$$

so that, if we use the facts that $\theta_{n_0} \leq \frac{\pi}{4}$ and that the function $x \mapsto x^2 + 1 - \sqrt{2}x$ is increasing on $[1, 2]$,

$$\frac{|a_{n_0} - b_{n_0}|}{m_{n_0}} = \sqrt{c_{n_0}^2 + 1 - 2c_{n_0} \cos \theta_{n_0}} \leq \sqrt{c_{n_0}^2 + 1 - \sqrt{2}c_{n_0}} \leq \sqrt{5 - 2\sqrt{2}}. \tag{10}$$

A direct induction using Equations (9) and (7) shows that, for all $k \geq 0$,

$$\log_2 \frac{|a_{n_0+k} - b_{n_0+k}|}{\alpha m_{n_0} \frac{\sin \theta_{n_0}}{\theta_{n_0}}} \leq 2^k \log_2 \frac{|a_{n_0} - b_{n_0}|}{\alpha m_{n_0} \frac{\sin \theta_{n_0}}{\theta_{n_0}}} \tag{11}$$

Using Equation (10) (and the fact that $\frac{\sin \theta_{n_0}}{\theta_{n_0}} \geq \frac{2\sqrt{2}}{\pi}$), we find that

$$\log_2 \frac{|a_{n_0} - b_{n_0}|}{\alpha m_{n_0} \frac{\sin \theta_{n_0}}{\theta_{n_0}}} \leq -2,$$

so that (by Equation (11)) for all $k \geq 0$,

$$\frac{|a_{n_0+k} - b_{n_0+k}|}{m_{n_0} \frac{\sin \theta_{n_0}}{\theta_{n_0}}} \leq \frac{\alpha}{2^{2^{k+1}}} \leq \frac{1}{2^{2^{k+1}-3}}.$$

We are now ready to prove the announced result: using Equation (5), one proves that for all $n \geq 0$, $|M(z) - a_n| \leq |a_n - b_n|$, and Equation (7) shows that $|M(z)| \geq m_n \frac{\sin\theta_n}{\theta_n}$ for all n, so that

$$\frac{|M(z) - a_{n_0+k}|}{|M(z)|} \leq \frac{|a_{n_0+k} - b_{n_0+k}|}{m_{n_0}\frac{\sin\theta_{n_0}}{\theta_{n_0}}} \leq \frac{1}{2^{2^{k+1}-3}}.$$

If we set $n_1 = n_0 + \lceil \log_2(N+3) - 1 \rceil$, then the above equation proves that a_{n_1} indeed is an approximation of $M(z)$ with relative precision N bits. □

This shows that Algorithm 3 can be used for the evaluation of $M(z)$.

Algorithm: `EvaluateM`
input : $z \in \mathcal{R}$, $N \in \mathbb{N}$
output: M such that $|M - M(z)| \,/\, |M(z)| < 2^{-N}$

$a \leftarrow 1$;
$b \leftarrow z$;
$K \leftarrow \lceil \log_2 |\log_2 |z|| \rceil + \lceil \log_2(N+3) \rceil$;
for $k = K-1$ ***to*** 0 **do**
  $c \leftarrow a + b$;
  $b \leftarrow \sqrt{ab}$;
  $a \leftarrow c/2$;
end
return a;

Algorithm 3: evaluation of the univariate AGM function M

Note that, using Newton iterations, square roots can be extracted to relative precision N bits in time $O(\mathcal{M}(N))$ [3].

We now discuss precision: it is easy to see that only a constant number of bits of relative precision can be lost at each iteration, hence it is sufficient to work with a relative precision of

$$N + 2 + 2n_N = O\left(N + \log|\log|z||\right)$$

bits, with n_N as defined in Proposition 12.

This leads to the following result:

Corollary 1. *The time complexity of algorithm* `EvaluateM` *is in*

$$O\left(\mathcal{M}\left(N + \log|\log|z||\right)\left(\log|\log|z|| + \log N\right)\right).$$

If we suppose that z lies in a compact subset of $\mathcal{R}$, then the complexity is in

$$O\left(\mathcal{M}(N)\log N\right).$$

6.2. **Evaluating k' at a** ***fixed*** **$\tau \in \mathcal{F}$.** In this section, we fix some $\tau \in \mathcal{F}$ and study the complexity of the evaluation of k' at τ as a function of the required relative precision N. Our algorithm is based on Theorem 3. Since precision considerations will play an important role in this algorithm and its complexity, we begin by investigating them, before describing the algorithm itself.

6.2.1. *Precision considerations.* We first note that if $k'(\tau)$ is needed to relative precision N bits, the remark from the beginning of Section 5 shows that it is sufficient to compute an approximation to absolute precision $N+4$ bits.

To evaluate k', we will use Newton iterations on the function f_τ. The validity of our algorithm will be based on the following (slight) variant of Theorem 3:

Proposition 13. *Let $\tau \in \mathcal{F}$ and let (z_n) and (z'_n) be two sequences such that*

$$|k'(\tau) - z_0| \leq \mathrm{Min}\left(\frac{1}{10F(\tau)}, \frac{1}{10G(\tau)}, \frac{|k'(\tau)|}{4}, \frac{|1-k'(\tau)|}{4}\right),$$

$z'_0 = z_0$ and for all $n \geq 1$:

$$|z_n - z'_n| \leq \left(\frac{1}{2}\right)^{2^n} |k'(\tau) - z_0|$$

and

$$z'_{n+1} = z_n + \frac{\pi\tau f_\tau(z_n)}{2g(z_n)}.$$

Then, for all $n \geq 0$, we have

$$|k'(\tau) - z_n| \leq \left(\frac{1}{2}\right)^{2^n-1} |k'(\tau) - z_0|\,.$$

Proof. The proof mimics that of Theorem 3. □

The interest of this proposition is that the sequence (z_n) used in Theorem 3 does converge to $k'(\tau)$, but cannot be computed exactly (since algorithms can only handle numbers with finite precision). Here, the elements of the sequence (z_n) approximate those of the sequence (z'_n), hence our algorithm will use the sequence (z_n).

If we define b_τ by

$$b_\tau = \log_2\left(\mathrm{Max}\left(10\,F(\tau), 10\,G(\tau), \frac{4}{|k'(\tau)|}, \frac{4}{|1-k'(\tau)|}\right)\right),$$

then Proposition 13 tells us that if z_0 is an approximation of $k'(\tau)$ to absolute precision b_τ bits and if, for all $n \geq 0$, z_{n+1} is an approximation of

$$z_n + \frac{\pi\tau f_\tau(z_n)}{2g(z_n)}$$

to absolute precision at least $2^{n+1} + b_\tau$ bits, then for all $n \geq 0$, z_n is an approximation of $k'(\tau)$ to absolute precision $2^n - 1 + b_\tau$ bits. Our algorithm will compute such a sequence (z_n) to approximate $k'(\tau)$.

We now study the precision at which computations have to be carried out to compute z_{n+1} from z_n: we have to compute

$$\frac{\pi\tau f_\tau(z_n)}{2g(z_n)}$$

to absolute precision $2^{n+1} + 1 + b_\tau$ bits (since one bit of precision can be lost when summing with z_n).

Since z_n is an approximation of $k'(\tau)$ with absolute precision higher than b_τ bits, then, using the definitions of b_τ, $F(\tau)$ and $G(\tau)$ and Taylor expansions as in the proof of Lemma 1, we get

$$\frac{-2}{\pi\tau} g(z_n) = \frac{df_\tau}{dz}\,(k'(\tau))\,(1+\varepsilon_g)$$

and

$$f_\tau(z_n) = (z_n - k'(\tau))\,\frac{df_\tau}{dz}\,(k'(\tau))\,(1+\varepsilon_f),$$

with $|\varepsilon_g| \leq 1/5$ and $|\varepsilon_f| \leq 1/5$, hence

$$\begin{aligned}\left|\frac{\pi\tau f_\tau(z_n)}{2g(z_n)}\right| &\leq 2\,|(k'(\tau) - z_n)| \\ &\leq \frac{1}{2^{2^n+b_\tau}}.\end{aligned}$$

Moreover, using the fact that

$$g\left(k'(\tau)\right) = \frac{1}{\theta_{01}^2(\tau)\theta_{10}^4(\tau)},$$

it is easy to prove (by considering the definitions of θ_{01} and θ_{10} as q-series and the fact that $\mathrm{Im}(\tau) \geq \sqrt{3}/2$ since $\tau \in \mathcal{F}$) that

$$\frac{1}{10\,|q|} \geq |g\left(k'(\tau)\right)| \geq \frac{1}{25}.$$

Taking into account the fact that a few bits of precision can be lost in arithmetic operations, this shows that $f_\tau(z_n)$ should be computed to absolute precision

$$2^{n+1} + 4 + b_\tau + \log_2\left(\frac{25\pi\,|\tau|}{2}\right) \leq 2^{n+1} + 10 + b_\tau + \log_2(|\tau|)$$

bits, and $g(z_n)$ to absolute precision

$$2^n + 4 + \log_2\left(\frac{1}{10\,|q|}\right) \leq 2^n + 1 - 4\mathrm{Im}(\tau)$$

bits or to relative precision $2^n + 3$ bits.

6.2.2. *The algorithm.* From the discussion above, and taking into account the few bits of precision that can be lost through arithmetic operations, Algorithm 4 can be used to evaluate $k'(\tau)$.

Algorithm: `Evaluatekp1`
input : $\tau \in \mathcal{F}$, $N \in \mathbb{N}$, $b_\tau \in \mathbb{N}$
output: k'_a such that $|k'(\tau) - k'_a| \,/\, |k'(\tau)| \leq 2^{-N}$

$z \leftarrow$`EvaluatekpNaive`(τ, b_τ);
$n \leftarrow 0$;
while $2^n - 1 + b_\tau \leq N + 4$ **do**
  $a \leftarrow$`EvaluateM`$(z,\ 2^{n+1} + 12 + b_\tau + \lceil\log_2(|\tau|)\rceil)$;
  $b \leftarrow$`EvaluateM`$(\sqrt{1-z^2},\ 2^{n+1} + 12 + b_\tau + 2\lceil\log_2(|\tau|)\rceil)$;
  $z \leftarrow z + \frac{\pi\tau z(1-z^2)(ia-\tau b)}{2a^3}$;
  $n \leftarrow n + 1$;
end
return z;

Algorithm 4: evaluation of k' using Newton iterations and the AGM

Since the precision at which the computations have to be carried out increases (it more or less doubles at each Newton iteration), the total running time for this algorithm is proportional to the running time for the last Newton iteration, where n will be on the order of $\log_2 N$. The cost of that last iteration is (using results from the last section on the complexity of the evaluation of the function M) in $O(\mathcal{M}(N)\log N)$. We thus have proven the following theorem:

Theorem 4. *If τ is fixed in $\mathcal{F}$, then there exists an algorithm that evaluates $k'(\tau)$ with relative precision N bits in time $O(\mathcal{M}(N)\log N) = O(N^{1+\epsilon})$.*

6.3. **Evaluating k' at any $\tau \in \mathcal{F}$.** In this section, we let τ vary in $\mathcal{F}$ and study the complexity of the evaluation of k' at τ as a function of τ and of the required relative precision N. This is mainly motivated by class polynomial computation, a natural application of our algorithms.

If we consider the algorithm introduced in the last section, then its running time when τ varies will now also depend on τ, and not only on the required precision N. In fact, two major problems arise when $\mathrm{Im}(\tau)$ increases:

- the value of b_τ increases as well (the study of the second order derivative of f_τ at $k'(\tau)$ shows that b_τ increases at least linearly with $\mathrm{Im}(\tau)$). This is not too important, since the naïve algorithm (Algorithm 2) takes a *constant* number of multiplications to evaluate k' at τ with a precision linear in $\mathrm{Im}(\tau)$, as remarked in Section 3.2.
- $k'(\tau)$ will tend to 1, hence $\sqrt{1-k'(\tau)^2} = k(\tau)$ will tend to 0. It is in fact equivalent to $4q^{1/2}$, and if we use algorithm `Evaluatekp1` (Algorithm 4) to evaluate $k'(\tau)$, then the computation of f_τ at each step will require the computation of $M(\sqrt{1-z_n^2})$, and since $\sqrt{1-z_n^2}$ is on the order of $k(\tau)$, this computation will require a number of AGM iterations in $O(\mathrm{Im}(\tau) + N)$, where N is the required precision.

Due to these reasons, the running time of the algorithm described above, if τ is *not* fixed in $\mathcal{F}$, is in $O(\mathcal{M}(N + \log \operatorname{Im}(\tau))(N + \log \operatorname{Im}(\tau)))$. We now consider a variant of that algorithm improving on its complexity.

The basic idea is that, if $k'(\tau)$ is known, then $\theta_{00}^2(\tau)$ can be computed as $\theta_{00}^2(\tau) = 1/M(k'(\tau))$, and since $k'(\tau) = \theta_{01}^2(\tau)/\theta_{00}^2(\tau)$, so can $\theta_{01}^2(\tau)$. Now, using the duplication formulae for the theta constants given in Section 2.2, $\theta_{00}^2(2^n\tau)$ and $\theta_{01}^2(2^n\tau)$ can also be computed, for all $n \geq 1$, using n AGM iterations. This gives an algorithm with running time in $O(\mathcal{M}(N+n)n)$ for the computation of $k'(2^n\tau)$ with precision N bits, from $k'(\tau)$.

We now define, for $r > 1$,

$$\mathcal{F}_r = \mathcal{F} \cap \{\tau \in \mathbb{C} : |\tau| \leq r\}.$$

For all $r > 1$, $\mathcal{F}_r$ is a compact subset of $\mathcal{F}$, and $F(\tau)$ and $G(\tau)$ are indeed bounded on $\mathcal{F}_r$, as a consequence of the following proposition:

Proposition 14. *Let U be an open subset of $\mathbb{C}$, h be an analytic function on U and K be a compact subset of U where h does not vanish. Then the function H defined on U by*

$$H(z) = \operatorname{Sup}_{n\geq 1} \left| \frac{h^{(n)}(z)}{n!h(z)} \right|^{\frac{1}{n}}$$

is bounded on K.

Proof. Since $K \subset U$, there exists $\varepsilon > 0$ such that for all $z \in K$ and $z' \in \mathbb{C}$,

$$|z' - z| \leq \varepsilon \Rightarrow z' \in U.$$

Then, by the Residue Theorem, for all $z \in K$ and $n \geq 1$,

$$h^{(n)}(z) = \frac{1}{2\pi i} \int_{C(z,\varepsilon)} \frac{h(t)}{(t-z)^{n+1}} dt,$$

where $C(z,\varepsilon)$ is the circle of radius ε centered at z. If we let h_m (resp. h_M) denote the maximum (resp. the minimum) of the function $|h(z)|$ on K, we then have

$$\left| \frac{h^{(n)}(z)}{n!h(z)} \right| \leq \frac{h_M}{n!\varepsilon^n h_m} \leq \frac{h_M}{\varepsilon^n h_m},$$

for all $n \geq 1$, hence

$$H(z) \leq \operatorname{Sup}_{n\geq 1} \frac{1}{\varepsilon} \left(\frac{h_M}{h_m} \right)^{\frac{1}{n}} \leq \frac{h_M}{\varepsilon h_m}.$$

$\square$

Since $F(\tau)$ and $G(\tau)$ are bounded on $\mathcal{F}_r$, there exists B_r such that for all $\tau \in \mathcal{F}_r$,

$$b_\tau \leq B_r.$$

Using Theorem 4, the above result implies that, for all $r > 1$, there exists an algorithm that evaluates $k'(\tau)$ with precision N bits for all $\tau \in \mathcal{F}_r$ and $N \in \mathbb{N}$ in running time $O(\mathcal{M}(N) \log N)$.

Now, if $\tau \in \mathcal{F}$, there exists $n \leq \operatorname{Im}(\tau)$ such that $\tau/2^n \in \mathcal{F}_2$, and $k'(\tau)$ can be computed either using the naïve algorithm if the required precision N is less than n (this takes time in $O(\mathcal{M}(N))$) or by first computing $k'(\tau/2^n)$ and then using n AGM iterations as seen above. This leads to Algorithm 5:

We thus have:

Theorem 5. *There exists an algorithm that, for all $N \in \mathbb{N}$, evaluates $k'(\tau)$ with relative precision N bits in time $O(\mathcal{M}(N) \log N) = O(N^{1+\epsilon})$ for all $\tau \in \mathcal{F}$.*

Note that a direct consequence of this result is that Hilbert class polynomials of degree h can be computed in time $O(h^{2+\epsilon})$ using root approximation and reconstruction, as described in [7].

Algorithm: `Evaluatekp2`
input : $\tau \in \mathcal{F}$, $N \in \mathbb{N}$, $N_2 \in \mathbb{N}$
output: k'_a such that $|k'(\tau) - k'_a| / |k'(\tau)| \leq 2^{-N}$

$n \leftarrow \lfloor \log_2(\mathrm{Im}(\tau)) \rfloor$;
if $N \leq 2n$ **then**
  return `EvaluatekpNaive`(τ, N);
end
$\tau' \leftarrow \tau/2^n$;
$c \leftarrow$`Evaluatekp1`$(\tau', N + 2\lceil \log_2(n) \rceil, B_2)$;
$d \leftarrow$`EvaluateM`$(c, N + 2\lceil \log_2(n) \rceil)$;
$b \leftarrow cd$;
$a \leftarrow 1/d$;
while $n > 0$ **do**
  $c \leftarrow a + b$;
  $b \leftarrow \sqrt{ab}$;
  $a \leftarrow c/2$;
  $n \leftarrow n - 1$;
end
return a/b;

Algorithm 5: evaluation of k' (variant)

7. Evaluating other functions

7.1. **Using modular polynomials.** It is a well known fact (see [5] for example) that if f_1 and f_2 are two modular functions for two subgroups (of finite index) Γ_1 and Γ_2 of Γ, then there exists a polynomial $\Phi_{f_1,f_2} \in \mathbb{C}[X,Y]$ such that, for all $\tau \in \mathcal{H}$,

$$\Phi_{f_1,f_2}\left(f_1(\tau), f_2(\tau)\right) = 0.$$

Such a polynomial is called a *modular polynomial*. As for the functions k' and j (for which a modular polynomial is in fact given by Equation (1)), it is often the case with "classical" modular functions that the modular polynomial linking them together can be chosen with coefficients in $\mathbb{Z}$, $\mathbb{Q}$ or some finite extension of $\mathbb{Q}$.

We now let f be a modular function for a subgroup Γ' of Γ, and we suppose that a modular polynomial $\Phi_{f,k'}$ linking f to k' is known to an arbitrary precision. Then for any $\tau \in \mathcal{F}$, f can be evaluated at τ by first computing the (univariate) polynomial $\Phi_{f,k'}(X, k'(\tau))$, and then by using Newton iterations on that polynomial. This yields an algorithm in $O(\mathcal{M}(N)\log N)$. The algorithm we gave to evaluate j is nothing but a (very) special case of the latter.

7.2. **Evaluating the Dedekind η function.** The Dedekind η function can be defined, for all $\tau \in \mathcal{H}$, by

$$\begin{aligned}\eta(\tau) &= q^{\frac{1}{12}} \prod_{n\geq 1}(1 - q^{2n}) \\ &= q^{\frac{1}{12}} \left(1 + \sum_{n\geq 1}(-1)^n (q^{n(3n-1)} + q^{n(3n+1)})\right).\end{aligned}$$

(The equivalence between these two definitions, known as Euler's Pentagonal Number Theorem, is proved for example in [14]).

Modular invariants defined using simple or double quotients of η functions are often used to compute class polynomials [9]. It is thus of interest to have a fast algorithm to evaluate η.

We first note that, using Theorem 2 and the very definition of k' ($k'(\tau) = \theta_{01}^2(\tau)/\theta_{00}^2(\tau)$), θ_{00} and θ_{01} can be evaluated at any $\tau \in \mathcal{F}$ in time $O(\mathcal{M}(N)\log N)$.

Now, using [16, pages 112–114], we have:

$$\theta_{00}(\tau) = \eta(\tau) f^2(\tau),$$

where f is a modular function satisfying the modular equation

$$f^{24} k'^2 (1 - k'^2) = 16.$$

We thus have

$$\eta^{12} = \frac{k'^2(1-k'^2)\theta_{00}^2}{16},$$

and it is clear that this equation can be used to evaluate η in time $O(\mathcal{M}(N) \log N)$ (the 12^{th} root can be extracted using Newton iterations).

8. Experimental results

8.1. **Precision needed in order to initialize Newton iterations on f_τ.** One of the problems is that the value of b_τ, which is an input to Algorithm 4 is not known (hence the value of B_τ, which is necessary for Algorithm 5 is not known either).

After having carried out a number of numerical experiments (see Section 8), we conjecture that, for all $\tau \in \mathcal{F}$,

$$b_\tau \le 100 + \pi \log_2(e) \mathrm{Im}(\tau) \le 100 + 4.6 \mathrm{Im}(\tau).$$

In order to give evidence for this conjecture, we computed the minimal precision at which the initial approximation of $k'(\tau)$ has to be computed so that the Newton iterations do indeed converge. Our experiments showed that that quantity mainly depends on $\mathrm{Im}(\tau)$ and not much on $\mathrm{Re}(\tau)$. Figure 1 shows how this minimal precision changes as $\mathrm{Im}(\tau)$ changes (while $\mathrm{Re}(\tau)$ is fixed, equal to 0.25). This data shows that the minimal precision increases roughly as $4.53\mathrm{Im}(\tau)$, which corresponds to $\log_2(|q|)$ (since $\log_2(|q|) = \log_2(\exp(-\pi\mathrm{Im}(\tau))) = \pi \log_2(e)\mathrm{Im}(\tau)$, and $\pi \log_2(e) \simeq 4.5324$). This is in accordance with our conjecture on the value of b_τ.

8.2. **Timings and comparison with the naïve algorithm.** We implemented algorithms `Evaluatekp1` and `EvaluatekpNaive` in C, using the libraries GMP [12], MPFR [13] and MPC [10] for the arithmetic of complex numbers.

For a fixed τ (here, $\tau = 0.123456789 + 1.23456789i$), the computation time for the evaluation of k' using our implementation of algorithm `Evaluatekp1` for a required precision up to one million bits (on an Athlon 64 running at 2.4 GHz) is pictured in Figure 2.

Figure 3 shows how our implementation of `EvaluatekpNaive` compares to that of `Evaluatekp1`, on the same machine as above.

Acknowledgements

I thank people from the INRIA *projet TANC*, particularly Andreas Enge and François Morain, for many fruitful discussions concerning this work.

References

[1] H. Baier and G. Köhler. How to compute the coefficients of the elliptic modular function $j(z)$. *Experiment. Math.*, 12(1):115–121, 2001.
[2] J. M. Borwein and P. B. Borwein. *Pi and the AGM*. John Wiley, 1987.
[3] R. P. Brent. *Analytic computational complexity*, chapter Multiple-precision zero-finding methods and the complexity of elementary function evaluation, pages 151–176. Academic Press, New York, 1975.
[4] D. A. Cox. The arithmetic-geometric mean of Gauss. *Enseign. Math.*, 30:275–330, 1984.
[5] M. Deuring. Die Klassenkörper der komplexen Multiplikation. In *Enzyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen*, volume Bd 1, H. 10, T. 2. Teubner, Stuttgart, 1958.
[6] R. Dupont. Borchardt's mean, theta constants in genus 2 and applications, 2005. In preparation.
[7] A. Enge. The complexity of class polynomial computation via floating point approximations, 2005. In preparation.
[8] A. Enge. The complexity of modular polynomial computation via floating point evaluation and interpolation, 2005. In preparation.
[9] A. Enge and F. Morain. Comparing invariants for class fields of imaginary quadratic fields. In C. Fieker and D. Kohel, editors, *Algorithmic Number Theory*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 252–266. Springer-Verlag, 2002. 5th International Symposium, ANTS–V, Sydney, Australia, July 7-12, 2002. Proceedings.

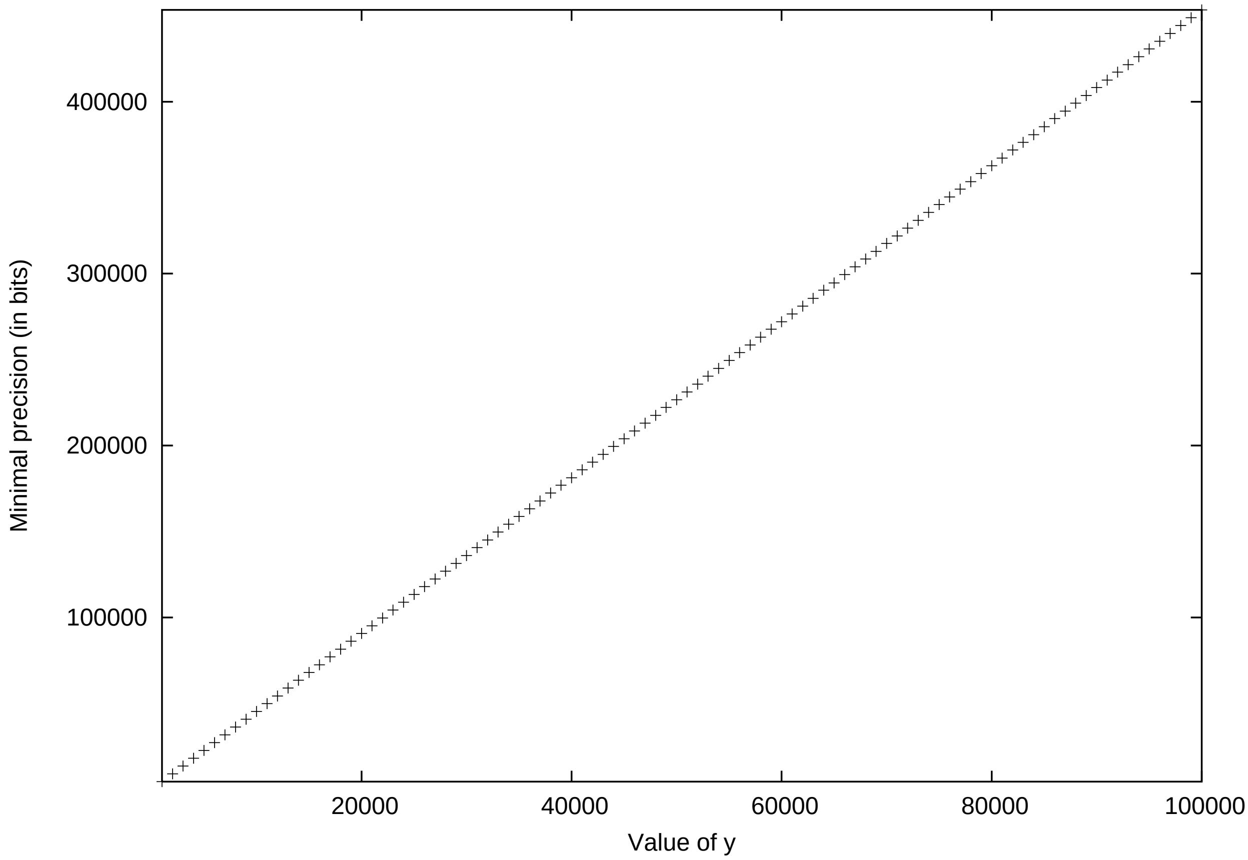

FIGURE 1. Minimal precision needed to initialize algorithm `Evaluatekp1` at $k'(0.25 + yi)$

[10] A. Enge and P. Zimmermann. MPC – Multiprecision Complex arithmetic library version 0.4, 2004. Available at `http://www.loria.fr/~zimmerma/free/`.
[11] C. F. Gauss. *Werke*. Dieterich, Göttingen, 1866–1933.
[12] T. Granlund *et al.* GMP – GNU Multiprecision library version 4.1, 2002. Available at `http://www.swox.com/gmp/`.
[13] G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR – Multiple Precision Floating point computations with exact Rounding library version 2.1.0, 2004. Available at `http://www.mpfr.org`.
[14] D. Mumford. *Tata lectures on theta I*. Birkhauser, 1984.
[15] B. Schoeneberg. *Elliptic modular functions*, volume 203 of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen*. Springer–Verlag, 1974.
[16] H. Weber. *Lehrbuch der Algebra*, volume III. Chelsea Publishing Company, New York, 1902.

INRIA FUTURS, PROJET TANC, LABORATOIRE LIX, ÉCOLE POLYTECHNIQUE, 91128 PALAISEAU, FRANCE
E-mail address: `dupont@lix.polytechnique.fr`

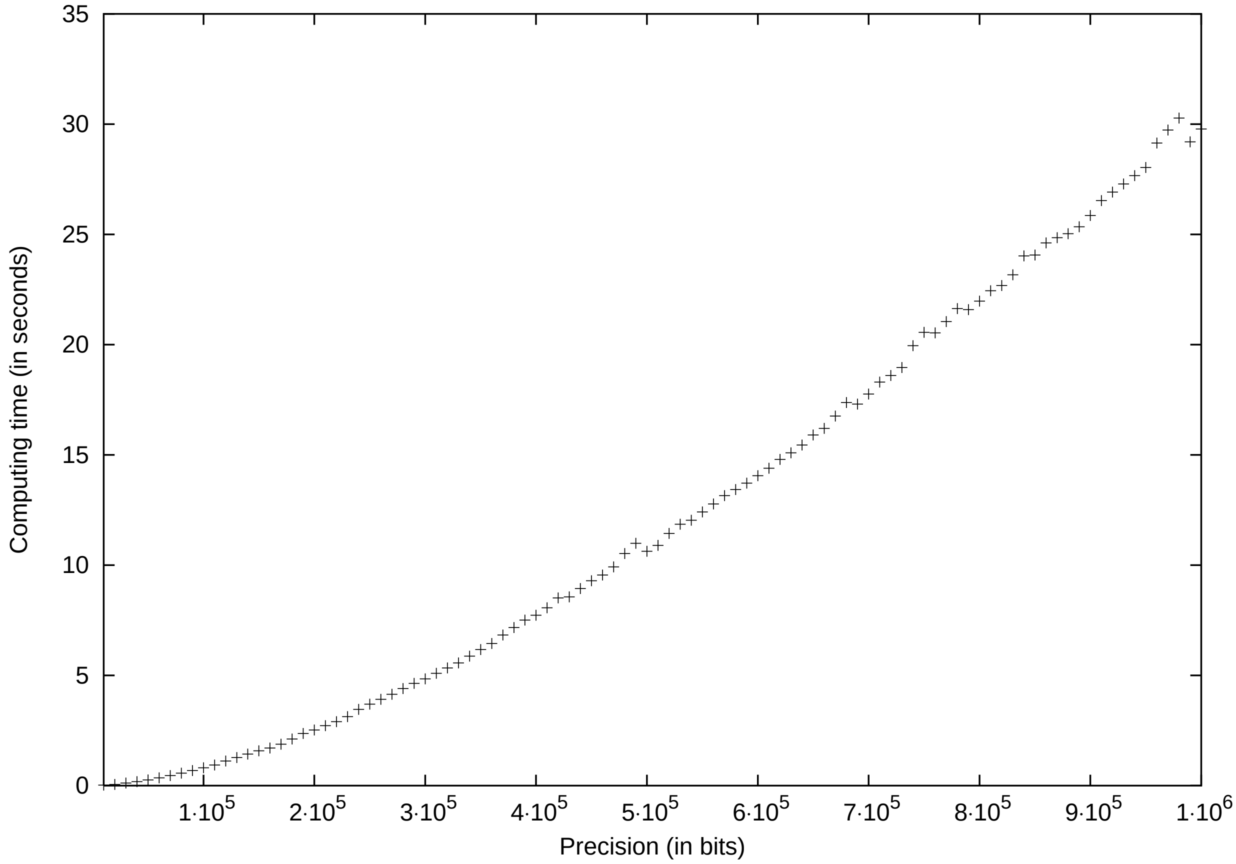

FIGURE 2. Computing time for algorithm `Evaluatekp1`

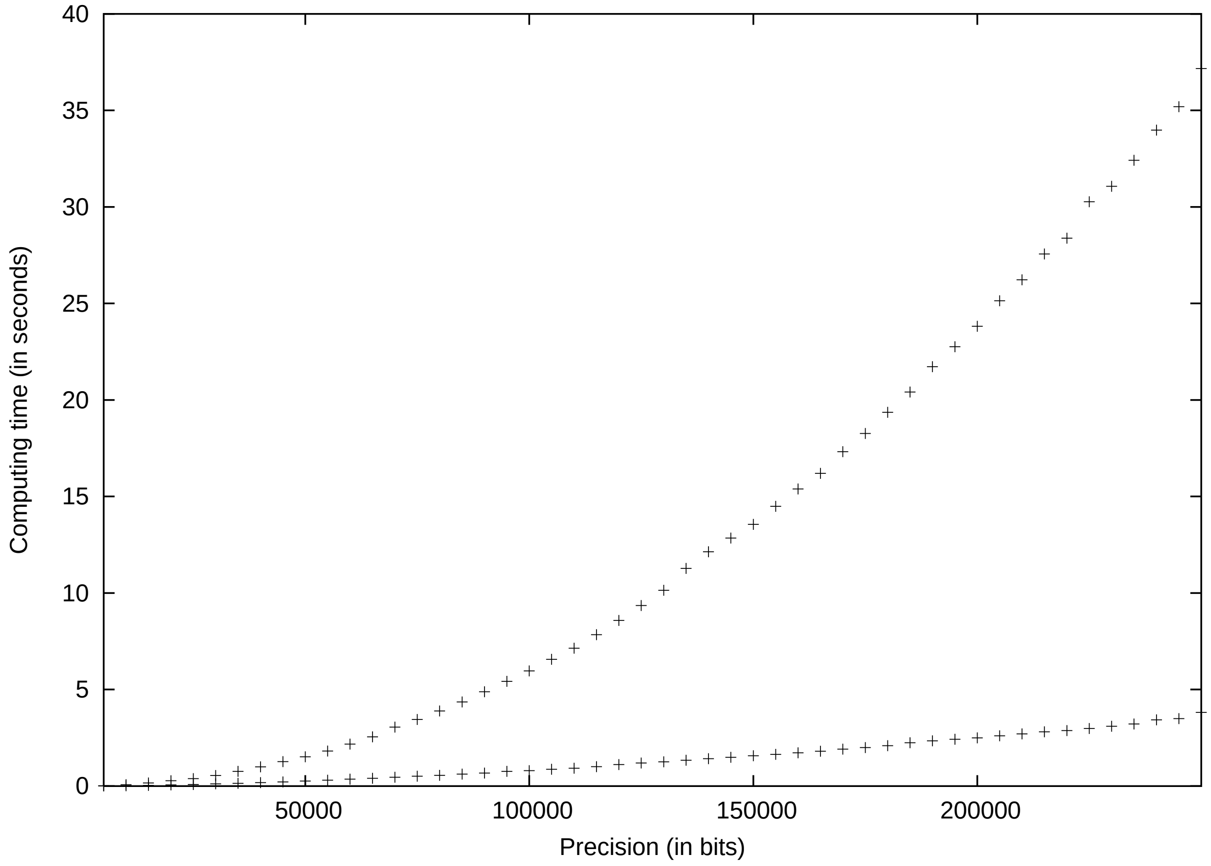

FIGURE 3. Computing time for algorithms `EvaluatekpNaive` (top) and `Evaluatekp1` (bottom)