Fast algorithms for computing isogenies between elliptic curves

Éric Schost, ORCCA, UWO

Joint work with Alin Bostan, François Morain, Bruno Salvy

Overview

Given two ℓ-isogeneous curves

$$E : y^2 = x^3 + Ax + B \quad \text{and} \quad E' : y'^2 = x'^3 + A'x' + B',$$

with additional conditions, we describe how to compute the isogeny $E \to E'$.

Main contribution: devising "asymptotically fast" versions of previous results (Atkin, Elkies), with essentially optimal complexity.

Applications, or potential applications: F. Morain's talk.

Outline

- review of algorithms for power series;
- application to isogeny computation.

Algorithms for power series

Basics

Basic fact: one can multiply polynomials fast.

Basics

Basic fact: one can multiply polynomials fast.

Let $\mathsf{M}(n)$ be such that polynomials of degree less than n can be multiplied in $\mathsf{M}(n)$ operations. Using FFT, one can take $\mathsf{M}(n)$ in $O(n \log n)$.

Basics

Basic fact: one can multiply polynomials fast.

Let $\mathsf{M}(n)$ be such that polynomials of degree less than n can be multiplied in $\mathsf{M}(n)$ operations. Using FFT, one can take $\mathsf{M}(n)$ in $O(n \log n)$.

Newton's iteration enables one to compute a power series solution $f \in k[[x]]$ of equations such as $\Phi(x, Y) = 0$, using the iteration

$$f_{i+1} = f_i - \frac{\Phi(x, f_i)}{\frac{\partial \Phi}{\partial Y}(x, f_i)} \bmod x^{2^{i+1}}.$$

Assuming that M satisfies

$$2\mathsf{M}(n) \leq \mathsf{M}(2n),$$

the cost is $O(\mathsf{M}(n))$.

Basics

Basic fact: one can multiply polynomials fast.

Let $\mathsf{M}(n)$ be such that polynomials of degree less than n can be multiplied in $\mathsf{M}(n)$ operations. Using FFT, one can take $\mathsf{M}(n)$ in $O(n \log n)$.

Newton's iteration enables one to compute a power series solution $f \in k[[x]]$ of equations such as $\Phi(x, Y) = 0$, using the iteration

$$f_{i+1} = f_i - \frac{\Phi(x, f_i)}{\frac{\partial \Phi}{\partial Y}(x, f_i)} \bmod x^{2^{i+1}}.$$

Assuming that M satisfies

$$2\mathsf{M}(n) \leq \mathsf{M}(2n),$$

the cost is $O(\mathsf{M}(n))$.

Many applications; what follows comes from the 1960's, 1970's and early 1980's [Sieveking, Kung, Brent, Schönhage].

Inversion [Cook-Sieveking-Kung 60's/70's]

Let f be in $k[[x]]$, with $f(0) \neq 0$. Then there exists g in $k[[x]]$ such that $fg = 1$.

Inversion [Cook-Sieveking-Kung 60's/70's]

Let f be in $k[[x]]$, with $f(0) \neq 0$. Then there exists g in $k[[x]]$ such that $fg = 1$.

1. Using coefficient extraction, the first n coefficients of g can be computed in $O(n^2)$ operations.

Remark: if f is a polynomial of degree d, the cost reduces to $O(nd)$.

Inversion [Cook-Sieveking-Kung 60's/70's]

Let f be in $k[[x]]$, with $f(0) \neq 0$. Then there exists g in $k[[x]]$ such that $fg = 1$.

1. Using coefficient extraction, the first n coefficients of g can be computed in $O(n^2)$ operations.

Remark: if f is a polynomial of degree d, the cost reduces to $O(nd)$.

2. Using Newton's iteration, the first n coefficients of g can be computed in $O(\mathsf{M}(n))$ operations, through the iteration

$$\left(g \bmod x^{2\ell}\right) = \left(g \bmod x^{\ell}\right)\left(2 - f(g \bmod x^{\ell})\right).$$

Remark: this formula comes from applying Newton's iteration to the operator

$$\Phi : \ g \mapsto \frac{1}{g} - f.$$

Logarithm and exponential [Brent'75]

Logarithm and exponential [Brent'75]

1. If g is in $1 + xk[[x]]$, its logarithm is defined as

$$\log(g) = -\sum_{i=1}^{n-1} \frac{1}{i}(1-g)^i.$$

By integrating g'/g, one can compute $\log(g) \bmod x^n$ in time $O(\mathsf{M}(n))$.

Logarithm and exponential [Brent'75]

1. If g is in $1 + xk[[x]]$, its logarithm is defined as

$$\log(g) = -\sum_{i=1}^{n-1} \frac{1}{i}(1-g)^i.$$

By integrating g'/g, one can compute $\log(g) \bmod x^n$ in time $O(\mathsf{M}(n))$.

2. Let f be in $k[[x]]$, with $f(0) = 0$. Its exponential is defined as

$$g = \exp(f) = \sum_{i=0}^{n-1} \frac{1}{i!} f^i.$$

Then $g \bmod x^n$ can be computed in time $O(\mathsf{M}(n))$ by the iteration

$$g_0 = 1, \quad g_{i+1} = g_i(1 + f - \log(g_i)) \bmod x^{2^{i+1}}$$

Logarithm and exponential [Brent'75]

1. If g is in $1 + xk[[x]]$, its logarithm is defined as

$$\log(g) = -\sum_{i=1}^{n-1} \frac{1}{i}(1-g)^i.$$

By integrating g'/g, one can compute $\log(g) \bmod x^n$ in time $O(\mathsf{M}(n))$.

2. Let f be in $k[[x]]$, with $f(0) = 0$. Its exponential is defined as

$$g = \exp(f) = \sum_{i=0}^{n-1} \frac{1}{i!} f^i.$$

Then $g \bmod x^n$ can be computed in time $O(\mathsf{M}(n))$ by the iteration

$$g_0 = 1, \quad g_{i+1} = g_i(1 + f - \log(g_i)) \bmod x^{2^{i+1}}$$

Remark: for these algorithms, one needs to invert $2, \ldots, n$ in k.

Application: conversions [Schönhage'82]

Application: conversions [Schönhage'82]

Let f be in $k[x]$ of degree n, write its factorization

$$f = \prod_{i \leq n} (x - x_i)$$

and let $p_j = \sum_{i \leq n} x_i^j$ be its power sums.

Application: conversions [Schönhage'82]

Let f be in $k[x]$ of degree n, write its factorization

$$f = \prod_{i \le n} (x - x_i)$$

and let $p_j = \sum_{i \le n} x_i^j$ be its power sums.

Let $\mathrm{rev}(f)$ be the reciprocal polynomial of f. Using

$$\mathrm{rev}(f) = \exp\left(-\sum_{i \ge 1} \frac{p_i}{i} x^i\right) \quad \text{or} \quad \frac{\mathrm{rev}(f)'}{\mathrm{rev}(f)} = -\sum_{i \ge 0} p_{i+1} x^i,$$

Application: conversions [Schönhage'82]

Let f be in $k[x]$ of degree n, write its factorization

$$f = \prod_{i \leq n} (x - x_i)$$

and let $p_j = \sum_{i \leq n} x_i^j$ be its power sums.

Let $\mathrm{rev}(f)$ be the reciprocal polynomial of f. Using

$$\mathrm{rev}(f) = \exp\left(-\sum_{i \geq 1} \frac{p_i}{i} x^i\right) \quad \text{or} \quad \frac{\mathrm{rev}(f)'}{\mathrm{rev}(f)} = -\sum_{i \geq 0} p_{i+1} x^i,$$

1. given f, one can recover $p_1, \dots, p_n$ in $O(\mathsf{M}(n))$ operations.

Application: conversions [Schönhage'82]

Let f be in $k[x]$ of degree n, write its factorization

$$f = \prod_{i \le n} (x - x_i)$$

and let $p_j = \sum_{i \le n} x_i^j$ be its power sums.

Let $\mathrm{rev}(f)$ be the reciprocal polynomial of f. Using

$$\mathrm{rev}(f) = \exp\left(-\sum_{i \ge 1} \frac{p_i}{i} x^i \right) \quad \text{or} \quad \frac{\mathrm{rev}(f)'}{\mathrm{rev}(f)} = -\sum_{i \ge 0} p_{i+1} x^i,$$

1. given f, one can recover $p_1, \ldots, p_n$ in $O(\mathsf{M}(n))$ operations.

2. given $p_1, \ldots, p_n$, one can recover f in $O(\mathsf{M}(n))$ operations, assuming one can invert $1, \ldots, n$ in k.

Application: conversions [Schönhage'82]

Let f be in $k[x]$ of degree n, write its factorization

$$f = \prod_{i \leq n} (x - x_i)$$

and let $p_j = \sum_{i \leq n} x_i^j$ be its power sums.

Let $\mathrm{rev}(f)$ be the reciprocal polynomial of f. Using

$$\mathrm{rev}(f) = \exp\left(-\sum_{i \geq 1} \frac{p_i}{i} x^i \right) \quad \text{or} \quad \frac{\mathrm{rev}(f)'}{\mathrm{rev}(f)} = -\sum_{i \geq 0} p_{i+1} x^i,$$

1. given f, one can recover $p_1, \ldots, p_n$ in $O(\mathsf{M}(n))$ operations.

2. given $p_1, \ldots, p_n$, one can recover f in $O(\mathsf{M}(n))$ operations, assuming one can invert $1, \ldots, n$ in k.

Classical algorithms: using Newton's relations, both conversions can be done in $O(n^2)$ operations.

Linear differential equations [Brent & Kung'78]

Let $f \in k[[x]]$ be a solution of the equation

$$af' + bf = c \quad \text{and} \quad f(0) = \alpha,$$

with a, b, c in $k[[x]]$.

Linear differential equations [Brent & Kung'78]

Let $f \in k[[x]]$ be a solution of the equation

$$af' + bf = c \quad \text{and} \quad f(0) = \alpha,$$

with a, b, c in $k[[x]]$.

Let

$$B = \frac{b}{a} \bmod x^{n-1} \quad \text{and} \quad C = \frac{c}{a} \bmod x^{n-1}.$$

Then,

$$f = \frac{1}{J}\left(\alpha + \int CJ\right) \bmod x^n$$

with $J = \exp(\int B)$.

Linear differential equations [Brent & Kung'78]

Let $f \in k[[x]]$ be a solution of the equation

$$af' + bf = c \quad \text{and} \quad f(0) = \alpha,$$

with a, b, c in $k[[x]]$.

Let

$$B = \frac{b}{a} \bmod x^{n-1} \quad \text{and} \quad C = \frac{c}{a} \bmod x^{n-1}.$$

Then,

$$f = \frac{1}{J}\left(\alpha + \int CJ\right) \bmod x^n$$

with $J = \exp(\int B)$.

Hence, $f \bmod x^n$ can be computed in $O(\mathsf{M}(n))$ operations, assuming one can invert $1, \ldots, n$ in k.

Remark: with Bostan, Chyzak, Ollivier, Salvy, Sedoglavic, extension to higher order.

Some non-linear differential equations [Brent & Kung'78]

Main idea: linearize the equation, and use the linear solver (this extends the idea behind Newton's iteration).

Some non-linear differential equations [Brent & Kung'78]

Main idea: linearize the equation, and use the linear solver (this extends the idea behind Newton's iteration).

Example: solving

$$(f')^2 = P(x, f)$$

with P in $k[[x]][Y]$.

Some non-linear differential equations [Brent & Kung'78]

Main idea: linearize the equation, and use the linear solver (this extends the idea behind Newton's iteration).

Example: solving

$$(f')^2 = P(x, f)$$

with P in $k[[x]][Y]$.

Assuming that the initial segment $f_0 = f \bmod x^\ell$ is known, write $f = f_0 + f_1$, with $f_1 = 0 \bmod x^\ell$. Then f_1 satisfies the linearized equation

$$2f_0' f_1' - \frac{\partial P}{\partial Y}(x, f_0) f_1 = P(z, f_0) - f_0'^2 \mod z^{2\ell-2},$$

Hence $f_1 \bmod x^{2\ell-1}$ can be computed in $O(\mathsf{M}(\ell))$ operations, assuming one can invert $1, \ldots, 2\ell$ in k.

Some non-linear differential equations [Brent & Kung'78]

Main idea: linearize the equation, and use the linear solver (this extends the idea behind Newton's iteration).

Example: solving

$$(f')^2 = P(x, f)$$

with P in $k[[x]][Y]$.

Assuming that the initial segment $f_0 = f \bmod x^\ell$ is known, write $f = f_0 + f_1$, with $f_1 = 0 \bmod x^\ell$. Then f_1 satisfies the linearized equation

$$2f_0' f_1' - \frac{\partial P}{\partial Y}(x, f_0) f_1 = P(z, f_0) - f_0'^2 \mod z^{2\ell-2},$$

Hence $f_1 \bmod x^{2\ell-1}$ can be computed in $O(\mathsf{M}(\ell))$ operations, assuming one can invert $1, \ldots, 2\ell$ in k.

Iterating the process, one can compute $f \bmod x^\ell$ in $O(\mathsf{M}(\ell))$ operations.

Some improvements

1. Middle product [Shoup'95, Hanrot *et al.*'02]

Newton's iteration involve recomputing known quantities.

Some improvements

1. Middle product [Shoup'95, Hanrot *et al.*'02]

Newton's iteration involve recomputing known quantities.

Example for inversion: if $g = 1/f \bmod x^\ell$,

$$2 - fg = 1 + 0 \cdot x + \cdots + 0 \cdot x^{\ell-1} + x^\ell R$$

so only R is needed. So in the product $\ell \times 2\ell$, which costs $2\mathsf{M}(\ell)$, only ℓ coefficients are needed.

Some improvements

1. Middle product [Shoup'95, Hanrot *et al.*'02]

Newton's iteration involve recomputing known quantities.

Example for inversion: if $g = 1/f \bmod x^\ell$,

$$2 - fg = 1 + 0 \cdot x + \cdots + 0 \cdot x^{\ell-1} + x^\ell R$$

so only R is needed. So in the product $\ell \times 2\ell$, which costs $2\mathsf{M}(\ell)$, only ℓ coefficients are needed.

General solution: the transposition principle shows that R can be computed in cost $\mathsf{M}(\ell)$, whatever M.

Some improvements

1. Middle product [Shoup'95, Hanrot *et al.*'02]

Newton's iteration involve recomputing known quantities.

Example for inversion: if $g = 1/f \bmod x^\ell$,

$$2 - fg = 1 + 0 \cdot x + \cdots + 0 \cdot x^{\ell-1} + x^\ell R$$

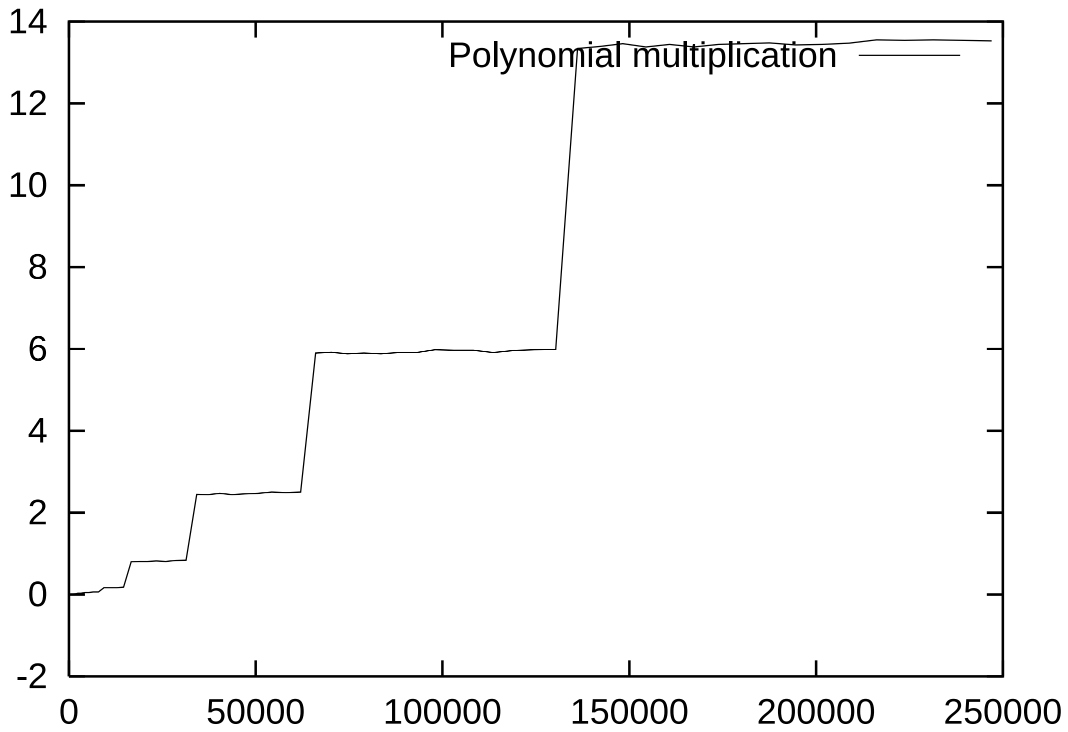
so only R is needed. So in the product $\ell \times 2\ell$, which costs $2\mathsf{M}(\ell)$, only ℓ coefficients are needed.

General solution: the transposition principle shows that R can be computed in cost $\mathsf{M}(\ell)$, whatever M.

Using FFT: instead of computing fg, it is enough to compute $fg \bmod x^{2\ell} - 1$, that is, to do FFT at 2ℓ points: the same as for a single product in degree ℓ.

Some improvements

2. Smoothing the multiplication. Using FFT multiplication, products are done on powers of 2 (at least, in most implementations). Hence the piecewise constant cost of multiplication.

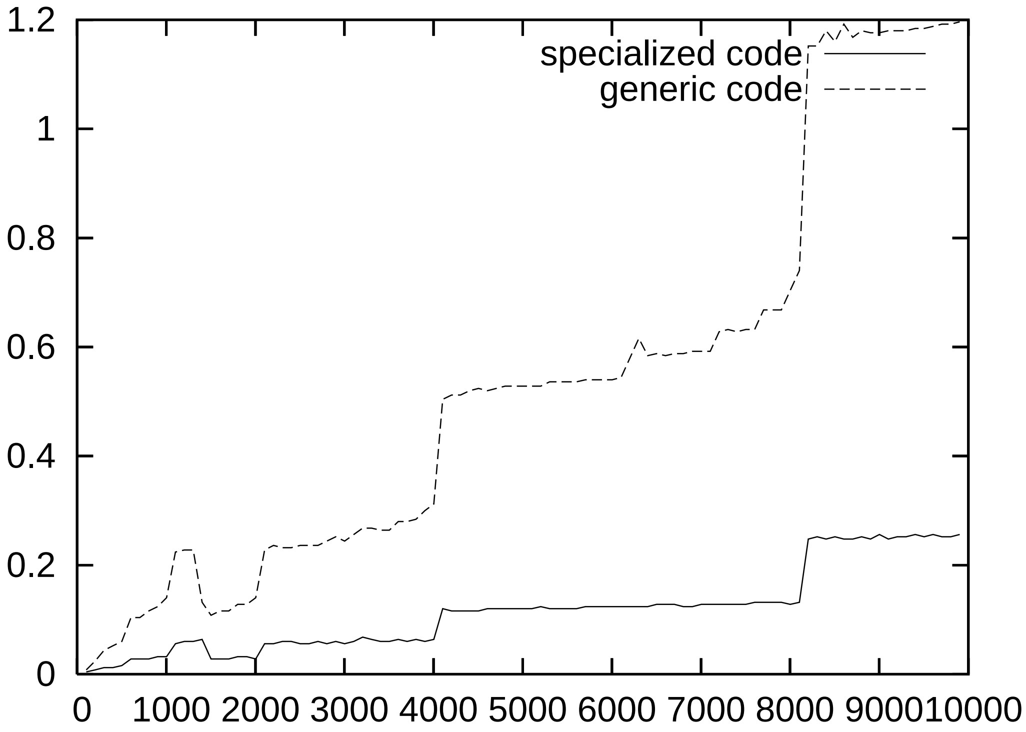


The Truncated Fourier Transform [van der Hoeven'04] smoothes the curves.

Some improvements

Bernstein gives lots of details on savings in Newton-like iterations.

Example: naive vs NTL's built-in inversion, modulo a 100 bit prime.



A remark on small characteristic

A priori, nothing works anymore in small characteristic because of the forbidden divisions, and non-uniqueness of the result.

A remark on small characteristic

A priori, nothing works anymore in small characteristic because of the forbidden divisions, and non-uniqueness of the result.

However, sometimes working at higher p-adic precision makes it possible to overcome some problems.

A remark on small characteristic

A priori, nothing works anymore in small characteristic because of the forbidden divisions, and non-uniqueness of the result.

However, sometimes working at higher p-adic precision makes it possible to overcome some problems.

Example: if f is in $\mathbb{F}_p[x]$, one cannot recover it from its power sums if $\deg f \geq p$.

A remark on small characteristic

A priori, nothing works anymore in small characteristic because of the forbidden divisions, and non-uniqueness of the result.

However, sometimes working at higher p-adic precision makes it possible to overcome some problems.

Example: if f is in $\mathbb{F}_p[x]$, one cannot recover it from its power sums if $\deg f \geq p$.

However, if one knows the power sums of a lift of f in $\mathbb{Z}/p^\alpha\mathbb{Z}\,[x]$, it is possible to recover f by applying Newton's iteration in $\mathbb{Z}/p^\alpha\mathbb{Z}\,[x]$.

A remark on small characteristic

A priori, nothing works anymore in small characteristic because of the forbidden divisions, and non-uniqueness of the result.

However, sometimes working at higher p-adic precision makes it possible to overcome some problems.

Example: if f is in $\mathbb{F}_p[x]$, one cannot recover it from its power sums if $\deg f \geq p$.

However, if one knows the power sums of a lift of f in $\mathbb{Z}/p^\alpha\mathbb{Z}\,[x]$, it is possible to recover f by applying Newton's iteration in $\mathbb{Z}/p^\alpha\mathbb{Z}\,[x]$.

A priori bounds: $\alpha = \log_p(\deg f!)$, since a precision $v_p(\deg f!)$ seems to be lost through the iterations.

A remark on small characteristic

A priori, nothing works anymore in small characteristic because of the forbidden divisions, and non-uniqueness of the result.

However, sometimes working at higher p-adic precision makes it possible to overcome some problems.

Example: if f is in $\mathbb{F}_p[x]$, one cannot recover it from its power sums if $\deg f \geq p$.

However, if one knows the power sums of a lift of f in $\mathbb{Z}/p^\alpha\mathbb{Z}\,[x]$, it is possible to recover f by applying Newton's iteration in $\mathbb{Z}/p^\alpha\mathbb{Z}\,[x]$.

A priori bounds: $\alpha = \log_p(\deg f!)$, since a precision $v_p(\deg f!)$ seems to be lost through the iterations.

With Bostan, Gonzàlez Vega, Perdry: it is enough to take $\alpha = \log_p(\deg f)$.

Going slightly further: similar results hold for first order differential equations

$$f' = af.$$

Rational reconstruction [Schönhage, Knuth, Moenck]

Let

$$f(x) = \frac{N(x)}{D(x)}$$

be a rational function, with $D(x) \neq 0$ and

$$\deg N(x), \deg D(x) \leq n.$$

Given the Taylor expansion of f at precision $2n$, one can reconstruct f using a Fast Euclidean Algorithm.

The complexity is $O(\mathsf{M}(n)\log(n))$ operations.

Rational reconstruction [Schönhage, Knuth, Moenck]

Let
$$f(x) = \frac{N(x)}{D(x)}$$
be a rational function, with $D(x) \neq 0$ and
$$\deg N(x), \deg D(x) \leq n.$$

Given the Taylor expansion of f at precision $2n$, one can reconstruct f using a Fast Euclidean Algorithm.

The complexity is $O(\mathsf{M}(n)\log(n))$ operations.

Remark: when
$$f(x) = \frac{D(x)'}{D(x)},$$
the fast exponentiation algorithm reduces the cost to $O(\mathsf{M}(n))$, if the characteristic is large enough.

Computing isogenies

Isogeneous curves

Let E, E' be given by

$$E : y^2 = x^3 + Ax + B \quad \text{and} \quad E' : y'^2 = x'^3 + A'x' + B'$$

and let

$$\Phi(x,y) = \big(\Phi_x(x,y), \Phi_y(x,y)\big) : \quad E \to E'$$

be an ℓ-isogeny. Then

$$\Phi(-P) = -\Phi(P)$$

implies

$$\Phi_x(x,y) = \varphi_x(x) \quad \text{and} \quad \Phi_y(x,y) = y\varphi_y(x).$$

Normalization

Let E be

$$E : y^2 = x^3 + Ax + B$$

and let F be a finite subgroup of E. Then the curve $E' = E/F$ is determined up to isomorphism.

Elkies's normalization: Write

$$E' : y'^2 = x'^3 + A'x' + B'.$$

Then we will work with the unique E' such that

$$\Phi : \quad E \to E'$$

is such that

$$\Phi^* \left(\frac{dx'}{y'} \right) = \frac{dx}{y}$$

(in general, $\Phi^* \left(\frac{dx'}{y'} \right) = \lambda \frac{dx}{y}$)

Normalization

Recall that $\Phi = (\varphi_x(x), y\varphi_y(x))$. Then

$$\Phi^* \left(\frac{dx'}{y'} \right) = \frac{\varphi_x'(x)}{y\varphi_y(x)} dx$$

so that

$$\varphi_x'(x) = \varphi_y(x)$$

In other words, Φ has the form

$$\Phi(x, y) = \big(\varphi(x), y\varphi'(x)\big).$$

From now on, we suppose that the normalized A', B' are given (e.g. from modular equations, cf. Elkies).

Remark: without the normalization,

$$\varphi(x) = \lambda\varphi'(x)$$

holds.

A first differential equation

Plugging the coordinates of Φ in the equation defining E' yields

$$(y\varphi(x)')^2 = \varphi(x)^3 + A'(\varphi(x)) + B'$$

that is

$$(x^3 + Ax + B)(\varphi(x)')^2 = \varphi(x)^3 + A'\varphi(x) + B'.$$

In particular, φ has the development

$$\varphi = x + \sum_i \frac{h_i}{x^i}.$$

We change the unknown function, defining

$$S(x) = \frac{1}{\sqrt{\varphi\left(\frac{1}{x^2}\right)}} \quad \text{or} \quad \varphi(x) = \frac{1}{S\left(\frac{1}{\sqrt{x}}\right)^2}$$

The regularized differential equation

Chain rule $\implies$ S satisfies the equation

$$(Bx^6 + Ax^4 + 1)\, S\,'(x)^2 = 1 + A'\, S(x)^4 + B'\, S(x)^6$$

with $S(x) = x + O(x^2)$. Hence, by

- computing the expansion of $C(x) = (Bx^6 + Ax^4 + 1)^{-1}$
- solving the equation $S'(x)^2 = C(x)(1 + A'S(x)^4 + B'S(x)^6)$,
- computing $\varphi(x) = \frac{1}{S\left(\frac{1}{\sqrt{x}}\right)^2}$

one can recover the first ℓ terms of the expansion of $\varphi(x)$ in $O(\mathsf{M}(\ell))$ operations.

Warning: this requires that the characteristic of k be large enough.

Then, φ can be recovered using rational function reconstruction in time $O(\mathsf{M}(\ell) \log(\ell))$.

Vélu's and Dewaghe's formulas

Let F be the kernel of Φ and let

$$D(x) = \prod_{Q \in F^*} (x - x_Q) = x^{\ell-1} - \sigma x^{\ell-2} + \sigma_2 x^{\ell-3} - \sigma_3 x^{\ell-4} + \cdots ,$$

Vélu's formulas imply that $\varphi(x)$ can be written

$$\varphi(x) = \frac{N(x)}{D(x)}$$

with

$$\begin{aligned} \frac{N(x)}{D(x)} &= \ell x - \sigma - (3x^2 + A)\frac{D'(x)}{D(x)} - 2(x^3 + Ax + B)\left(\frac{D'(x)}{D(x)}\right)' \\ &= \ell x - \sigma - 2\sqrt{x^3 + Ax + B}\left(\sqrt{x^3 + Ax + B}\,\frac{D'(x)}{D(x)}\right)' \end{aligned}$$

(easy) extension of [Dewaghe 99] to arbitrary ℓ.

Application: speeding-up the reconstruction

Write

$$\varphi(x) = \frac{N(x)}{D(x)} = x + \sum_{i \geq 1} \frac{h_i}{x^i}$$

and $p_1 = \sigma, p_2, \ldots$ be the power sums of D.

Identifying coefficients in Dewaghe's formula yields

$$h_i = (2i+1)p_{i+1} + (2i-1)Ap_{i-1} + (2i-2)Bp_{i-2}, \quad \text{for all } i \geq 1.$$

Then, knowing the first ℓ coefficients h_i, one can recover the first ℓ power sums of D in $O(\mathsf{M}(\ell))$ operations.

Warning 1: this requires that the characteristic of k be large enough.

Warning 2: this requires that the first power sum σ be known (this can be deduced from modular equations).

Previous work

1. Elkies 1992 and 1998: $O(\ell^2)$ algorithms. Our algorithm is a fast version of Elkies 1998.

2. Atkin 1992, $O(\ell^2)$ algorithm based on the equation

$$D(\wp(x)) = \exp\left(-\sigma x^2 + 2\iint \ell\wp(x) - \tilde{\wp}(x)\right).$$

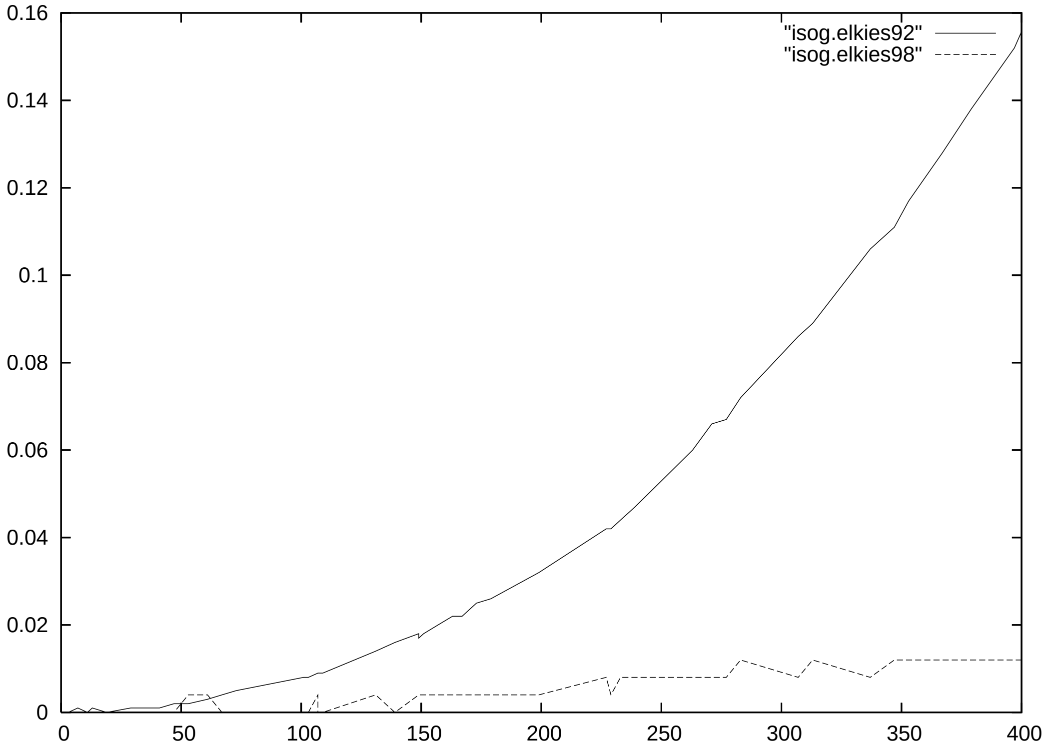
where $\wp$ and $\tilde{\wp}$ are the Weiestrass functions on E and E'.

Using Dewaghe's equation, and writing

$$\varphi = \tilde{\wp} \circ \wp^{-1},$$

this method is seen to be equivalent to Elkies'.

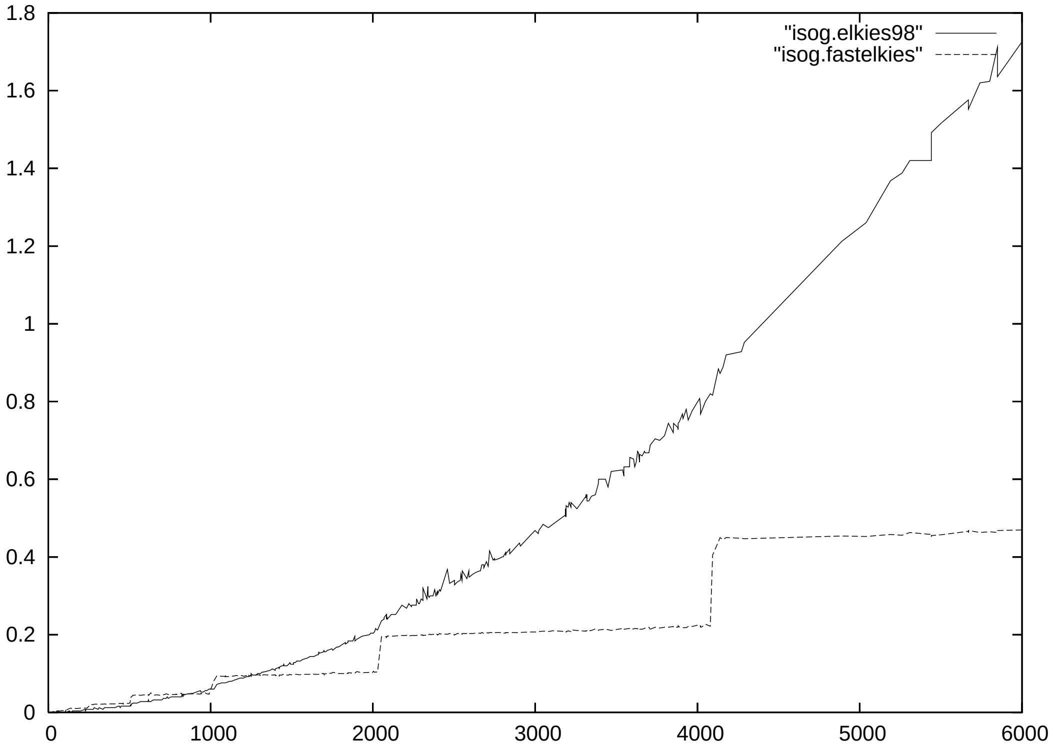
Results



Elkies 1992 vs Elkies 1998 over $\mathbb{F}_p$, with $p = 10^{19} + 51$.

We are comparing two $O(\ell^2)$ algorithms. The difference is in the constant.

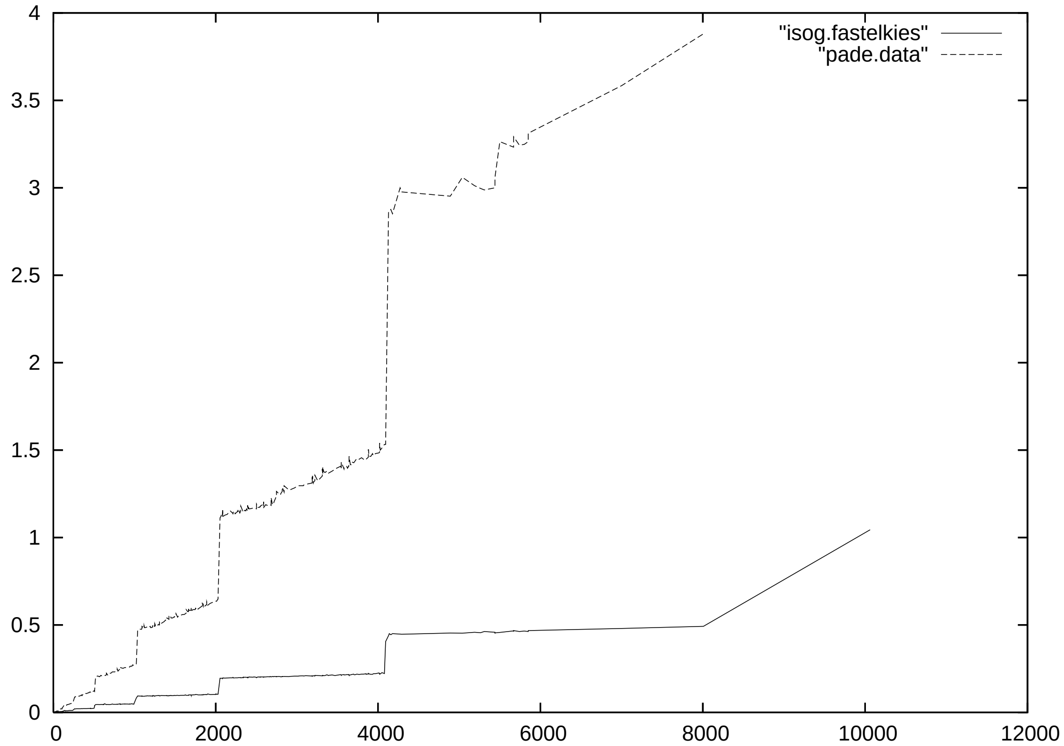
Results



Elkies 1998 vs fast Elkies over $\mathbb{F}_p$, with $p = 10^{19} + 51$.

Crossover is in the range 100-200, but our implementation incorporates none of the fancy tricks seen before.

Results



Elkies 1998: rational reconstruction vs exponential over $\mathbb{F}_p$, with $p = 10^{19} + 51$.

Rational reconstruction is much slower than fast exponentiation, even though differing by only a log factor.

Conclusion, further work

1. What's new: Introduction of fast arithmetic (but no refined optimization.

2. Still missing 2: small characteristic case.