COMPUTING THE CHARLAP-COLEY-ROBBINS MODULAR POLYNOMIALS

FRANÇOIS MORAIN

ABSTRACT. Let $\mathcal{E}$ be an elliptic curve over a field $\mathbf{K}$ and ℓ a prime. There exists an elliptic curve $\mathcal{E}^*$ related to $\mathcal{E}$ by an isogeny (rational map that is also a group homomorphisms) of degree ℓ if and only $\Phi_\ell(X, j(\mathcal{E})) = 0$, where $\Phi_\ell(X, Y)$ is the traditional modular polynomial. Moreover, the modular polynomial gives the coefficients of $\mathcal{E}^*$, together with parameters needed to build the isogeny explicitly. Since the traditional modular polynomial has large coefficients, many families with smaller coefficients can be used instead, as described by Elkies, Atkin and others. In this work, we concentrate on the computation of the family of modular polynomials introduced by Charlap, Coley and Robbins. It has the advantage of giving directly the coefficients of $\mathcal{E}^*$ as roots of these polynomials. We review and adapt the known algorithms to perform the computations of modular polynomials. After describing the use of series computations, we investigate fast algorithms using floating point numbers based on fast numerical evaluation of Eisenstein series. We also explain how to use isogeny volcanoes as an alternative.

1. INTRODUCTION

Computing isogenies is the central ingredient of the Schoof-Elkies-Atkin (SEA) algorithm that computes the cardinality of elliptic curves over finite fields of large characteristic [38, 1, 20] and also [6]. More recently, it has found its way in postquantum cryptography [14, 26, 11, 25] among others, as well as the cryptosystems [17, 37, 24].

Let $\mathbf{K}$ be a field of characteristic different from 2 and 3. An isogeny between two elliptic curves $\mathcal{E}/\mathbf{K} : Y^2 = X^3 + AX + B$ and $\mathcal{E}^*/\mathbf{K} : Y^2 = X^3 + A^*X + B^*$ is a group morphism that is a rational map of degree ℓ (the cardinality of its kernel). There are two ways to handle these isogenies. When the degree ℓ is small, formulas for A^*, B^* and the kernel polynomial can be precomputed. For large ℓ, one of the key ingredients is modular polynomials, the second one finding rational expressions for A^* and B^* from A, B and the modular polynomial. We concentrate here on the former problem. The second is treated in [32]. Note there is a purely algebraic approach using triangular sets [35, §7].

There are many families of modular polynomials that can be used, with different properties. Very generally, a modular polynomial is some bivariate

Date: February 9, 2023.

polynomial $\Phi(X, J)$ where J corresponds to the j-invariant of the elliptic curve $\mathcal{E}$, and X stands for some modular function on $\Gamma_0(\ell)$. The prototype is $j(\mathcal{E}^*)$ (see below for more precise statements) that yields traditional modular polynomials. Alternative choices for X exist. They all lead to polynomials of (conjectured) height $O((\ell+1)\log \ell)$ but with small constants.

The Charlap-Coley-Robbins (CCR) modular polynomials [12] offer an alternative as a triplet of polynomials (U_ℓ, V_ℓ, W_ℓ) having the property (among others) that A^* (resp. B^*) is a root of V_ℓ (resp. W_ℓ). They can be used in conjunction with the preceding modular polynomials (in the case where some partial derivative vanishes).

The aim of this work is describe the properties of CCR polynomials (Section 4), explain how to compute them (Section 5) and use them for isogeny computations. We adapt methods from the classical approach: series expansions over $\mathbb{Z}$ or $\mathbb{F}_p$ for small primes and Chinese remaindering theorem, evaluation/interpolation with floating point numbers, isogeny volcanoes. For doing this, we need to evaluate Eisenstein series (introduced in Section 2) at high precisions and we describe algorithms to do so in Section 3, including an approach to the simultaneous evaluation of several series. We give numerical examples and height comparisons between modular polynomials. We finish with the computation of algebraic expressions for A^* and B^* as rational fractions (see [33]).

2. Classical functions

Put $q_1 = \exp(i\pi\tau)$ and $q = q_1^2$. Depending on authors, formulas are expressed in either parameters, which sometimes is clumsy. We write indifferently $f(\tau)$ or $f(q)$ some series.

2.1. Jacobi θ functions.

The classical θ functions are:

$$\theta_2(q_1) = \sum_{n\in\mathbb{Z}} q_1^{(n+1/2)^2},\ \theta_3(q_1) = \sum_{n\in\mathbb{Z}} q_1^{n^2},\ \theta_4(q_1) = \sum_{n\in\mathbb{Z}} (-1)^n q_1^{n^2}.$$

Among many properties, one has

$$\theta_3^4(q_1) = \theta_4^4(q_1) + \theta_2^4(q_1).$$

The latter formula enables to concentrate on the evaluation of $\theta_{3,4}(q_1)$ as is done in [18].

Note also the following [18, Prop. 4]

Proposition 2.1.

$$\lim_{\mathrm{Im}(\tau)\to+\infty} \theta_3(\tau) = 1,\ \lim_{\mathrm{Im}(\tau)\to+\infty} \theta_4(\tau) = 1,\ \lim_{\mathrm{Im}(\tau)\to+\infty} \theta_2(\tau) = 0.$$

2.2. Eisenstein series.

2.2.1. *Definitions, first properties.* The classical Eisenstein series[1] we consider are

$$E_2(q) = 1 - 24\sum_{n=1}^{\infty}\sigma_1(n)q^n,$$

$$E_4(q) = 1 + 240\sum_{n=1}^{\infty}\sigma_3(n)q^n,$$

$$E_6(q) = 1 - 504\sum_{n=1}^{\infty}\sigma_5(n)q^n,$$

where $\sigma_r(n)$ denotes the sum of the r-th powers of the divisors of n. Other series E_{2k} can be defined for $k > 3$. Theory tells us that E_{2k} is a modular form of weight $2k$ for $k > 1$. As a result, a series for $k > 3$ can be expressed as polynomials in E_4 and E_6. Also of interest is the discriminant Δ:

$$\Delta(q) = (E_4(q)^3 - E_6(q)^2)/1728 = \eta(q)^{24}$$

with $\eta(q) = q\prod_{n=1}^{\infty}(1-q^n)$ is the Dedekind function. Finally, the modular invariant is

$$j(q) = \frac{E_4(q)^3}{\Delta(q)} = \frac{1}{q} + 744 + \cdots.$$

The quantities (see [23, §13.20])

$$a = \theta_2(q_1), b = \theta_3(q_1), c = \theta_4(q_1)$$

satisfy the following identities (among others)

$$(1)\quad E_4 = (a^8+b^8+c^8)/2,\ E_6 = (a+b)(b+c)(c-a)/2,\ \Delta = (abc/2)^8.$$

From which we deduce

$$\lim_{\mathrm{Im}(\tau)\to+\infty} E_4(\tau) = 1,\ \lim_{\mathrm{Im}(\tau)\to+\infty} E_6(\tau) = 1.$$

2.2.2. *The special case of E_2.* The series E_2 is not a modular form since (see [36]):

Theorem 2.2. *For all matrices* $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ *of* $\mathrm{SL}_2(\mathbb{Z})$*, one has*

$$E_2((a\tau+b)/(c\tau+d)) = (c\tau+d)^2E_2(\tau) + \frac{6c}{\pi i}(c\tau+d).$$

We can build a modular form easily as follows, using Ramanujan's multipliers. Let n be an integer and Let $\mathcal{F}_n$ denote the *multiplier* $E_2(\tau)-nE_2(n\tau)$.

Proposition 2.3. *The function $\mathcal{F}_n$ is a modular form of weight 2 and trivial multiplier system for $\Gamma_0(n)$.*

[1]Ramanujan used $L = P = E_2$, $M = Q = E_4$, $N = R = E_6$.

Proof: Write, for $ad-bc=1$ the value

$$E_2\left(n\frac{a\tau+b}{c\tau+d}\right) = E_2\left(\frac{a(n\tau)+nb}{(c/n)(n\tau)+d}\right)$$

$$= ((c/n)n\tau+d)^2E_2(n\tau) - \frac{6c}{n\pi i}((c/n)n\tau+d)$$

which leads to

$$nE_2\left(n\frac{a\tau+b}{c\tau+d}\right) = n(c\tau+d)^2E_2(n\tau) - \frac{6c}{\pi i}(c\tau+d).$$

Subtracting $E_2((a\tau+b)/(c\tau+d))$, we see that

$$\mathcal{F}_n((a\tau+b)/(c\tau+d)) = (c\tau+d)^2\mathcal{F}_n(\tau).\square$$

Some identities are known for small values of n, for instance [29] for $n\in\{2,4\}$; [2] for $n=3$ and 11; [7, Thm 6.2], [3, Thm 3.7] for $n\in\{5,7\}$. A very nice relation is [3, Thm 6.3]

$$\mathcal{F}_7(q) = 6\left(\sum_{m,n=-\infty}^{\infty} q^{m^2+mn+2n^2}\right)^2.$$

3. Fast evaluations

3.1. **Fast evaluation of $E_{2k}(q)$ for $k\geq 1$.** The θ functions that can be evaluated at precision N in time $O(\mathsf{M}(N)\sqrt{N})$ with q_1-expansions (see [22]) or faster in $O(\mathsf{M}(N)\log N)$ using [18] and also [30]. It follows that the quantities E_{2k} (for $k\geq 2$) can be evaluated at precision N in $O(\mathsf{M}(N)\log N)$ operations. As a consequence $j(q)$ can also be evaluated with the same complexity.

Evaluating E_2 is less obvious. However, hidden in the proof of [28, Thm 4] (thanks to [29] for highlighting this), we find

$$\frac{E_2E_4}{E_6} = \frac{{}_2F_1\left(\frac{13}{12},\frac{5}{12};1;\frac{1728}{j}\right)}{{}_2F_1\left(\frac{1}{12},\frac{5}{12};1;\frac{1728}{j}\right)} = 1+\frac{720}{j}+\cdots$$

where the Gauss hypergeometric function is defined by

$${}_2F_1\left(a,b;c;x\right) = \sum_{k=0}^{\infty}\frac{(a)_k(b)_k}{(c)_k\,k!}x^k, |x|<1$$

where $(a)_k = a(a+1)\cdots(a+k-1)$. By [40, 41, 31] and also [9], this function can be computed at precision N in $O(\mathsf{M}(N)(\log N)^2)$ operations. See also [27] for realistic computations.

Other links with hypergeometric functions could be investigated (A .Bostan, personal communication).

Also, note that actually we need to evaluate $\mathcal{F}_\ell$ for prime ℓ. Sometimes, we may use special formulas as indicated above.

3.2. **A multi-value approach.** In practice, a simpler approach yields the values E_{2k} of many k's with $k \geq 1$ in time $O(\mathsf{M}(N)\sqrt{N})$ based on [22]. The cost reduces to that of one series evaluation.

From [4], we take

$$(q;q)_\infty = \exp(-2i\pi\tau/24)\eta(q),$$

and for $k \geq 0$:

$$T_{2k}(q) = 1 + \sum_{n=1}^{\infty}(-1)^n \left\{(6n-1)^{2k} q^{n(3n-1)/2} + (6n+1)^{2k} q^{n(3n+1)/2}\right\}.$$

Note that $(q;q)_\infty = T_0(q)$.

Theorem 3.1 (Section 6)**.**

$$\frac{T_2(q)}{T_0(q)} = E_2,\ \frac{T_4(q)}{T_0(q)} = 3E_2^2 - 2E_4,\ \frac{T_6(q)}{T_0(q)} = 15E_2^3 - 30E_2E_4 + 16E_6.$$

General formulas for $T_{2k}(q)/T_0(q)$ are given in the reference.

If we need to compute E_2, E_4 and E_6, we see that it is enough to evaluate the series T_{2k} for $k \in \{0,1,2,3\}$ followed by a handful of multiplications and divisions as given in the preceding Theorem. Moreover, we can evaluate these series by sharing the common powers of q. These powers are evaluated at a reduced cost using [22, Algorithm2]. We give the modified procedure as algorithm 1. In Step 3.2.3, we have added the contribution $(6n \pm 1)^{2i}$ to each `T[i]`. We assume that the cost of multiplying by these small quantities is negligible. Were it not the case, we could use incremental computations of the polynomials $(6n \pm 1)^{2i}$. The cost of this algorithm reduces to that of one of the series, gaining a factor $kmax$.

Algorithm 1 uses the primitive in Algorithm 2. The reason of Step 4 is that $T[k]$ will be close to 1 when q is small, so that we may not want to add 1 right at the beginning and perhaps not in this function.

3.3. **The case of imaginary arguments.** In practice, it is easier to consider $\tau = \rho i$ for real $\rho \geq 1$. In that case, $1 > q_0 = \exp(-2\pi) \geq q = \exp(-2\pi\rho) > 0$. The functions E_2 and E_6 are increasing from $E_{2k}(q_0)$ to 1 (note that $E_6(q_0) = 0$ and $E_2(q_0) = 3/\pi$ from [19]); E_4 is decreasing from $E_4(q_0)$ to 1. This is important to note for the computations not to explode. Remember also that $j(i) = 1728$.

We turn to the precision needed for evaluating the functions T_{2k}. Let N denote an integer and $T_{2k,N}$ the truncated sum up to $n = N-1$. Since the series is alternating, we can bound the error using

$$\begin{aligned}&|T_{2k}(q) - T_{2k,N}(q)|\\ &\leq \left\{(6N-1)^{2k}\, q^{N(3N-1)/2} + (6N+1)^{2k}\, q^{N(3N+1)/2}\right\}\\ &\leq \left((6N-1)^{2k} + (6N+1)^{2k}\right) q^{N(3N-1)/2}.\end{aligned}$$

Since $0 < q < 1$, this gives us a quadratic convergent series.

Algorithm 1: Combined evaluation of $T_{2k}(q)$.

Function *EvaluateManyT(q, N, kmax)*
Input : q, N, $kmax$
Output: $(T_{2k}(q))$ for $0 \le k \le kmax$
1. **for** $k := 0$ **to** $kmax$ **do**
  $T[k] \leftarrow 0$
2. $s \leftarrow 1$; $A \leftarrow \{1\}$; $Q[1] \leftarrow \{q\}$; $c \leftarrow 0$
3. **for** $n := 1$ *while* $n(3n+1)/2 \le N$ **do**
  $s \leftarrow -s$
  // $s = (-1)^n$
  3.1 $c \leftarrow c + 2n - 1$
  3.2 **for** $r := 1$ **to** 2 **do**
    3.2.1 **if** $r = 2$ **then**
      $c \leftarrow c + n$
      // $c = n(3n+1)/2$
    3.2.2 $q' \leftarrow$ FindPowerInTable(A, Q, c)
    3.2.3 $C \leftarrow (6n + (-1)^r)^2$
    3.2.4 **for** $k := 0$ **to** $kmax$ **do**
      $T[k] \leftarrow T[k] + sq'$
      if $k < kmax$ **then**
        $q' \leftarrow Cq'$
4. **for** $k := 0$ **to** $kmax$ **do**
  $T[k] \leftarrow T[k] + 1$
5. **return** T.

4. The polynomials of Charlap-Coley-Robbins

4.1. **Division polynomials.** For $\mathcal{E} : y^2 = x^3 + Ax + B$, multiplication of a point (X, Y) by positive n on $\mathcal{E}$ is given by

$$[n](X,Y) = \left(\frac{\phi_n(X,Y)}{\psi_n(X,Y)^2}, \frac{\omega_n(X,Y)}{\psi_n(X,Y)^3} \right)$$

where the polynomials satisfy

$$\phi_n = X\psi_n^2 - \psi_{n+1}\psi_{n-1},\ 4Y\omega_n = \psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2$$

and $\phi_n, \psi_{2n+1}, \psi_{2n}/(2Y), \omega_{2n+1}/Y, \omega_{2n}$ belong to $\mathbb{Z}[A, B, X]$. It is customary to simplify this using

$$f_n(X) = \begin{cases} \psi_n(X,Y) & \text{for } n \text{ odd} \\ \psi_n(X,Y)/(2Y) & \text{for } n \text{ even} \end{cases}$$

with first values

$$f_{-1} = -1,\ f_0 = 0,\ f_1 = 1,\ f_2 = 1,$$

Algorithm 2: Finding c as a combination of known values.

Function *FindPowerInTable(A, Q, c)*
- **Input :** $A = \{a_1, \ldots, a_z\}$, Q such that for all i, $Q[a_i] = q^{a_i}$, c
- **Output:** q^c; A and Q are updated
- **if** $c = 1$ **then**
  - $q' \leftarrow Q[1]$
- **else if** $c = 2a$ *with* $a \in A$ **then**
  - $q' \leftarrow Q[a]^2$
- **else if** $c = a + b$ *with* $a, b \in A$ **then**
  - $q' \leftarrow Q[a] \cdot Q[b]$;
- **else if** $c = 2a + b$ *with* $a, b \in A$ **then**
  - $q' \leftarrow Q[a]^2 \cdot Q[b]$
- $A \leftarrow A \cup \{c\}$
- $Q[c] \leftarrow q'$
- **return** q'.

$$f_3(X, Y) = 3X^4 + 6AX^2 + 12BX - A^2$$

$$f_4(X, Y) = X^6 + 5AX^4 + 20BX^3 - 5A^2X^2 - 4ABX - 8B^2 - A^3.$$

The degree of f_n is $(n^2 - 1)/2$ for odd n and $(n^2 - 4)/2$ for even n. If X has weight 1, A weight 2 and B weight 3, all monomials in f_n have the same weighted degree equal to the degree of f_n.

4.2. **The work of Elkies.** An isogeny is associated with its kernel, or its polynomial description (called *kernel polynomial*). Given some finite subgroup F of $\mathcal{E}$, one can build an isogenous curve $\mathcal{E}^*$ and the corresponding isogeny, using Vélu's formulas. In the context of point counting, we discover a curve $\mathcal{E}^*$ that is ℓ-isogenous to $\mathcal{E}$ via its j-invariant as a root of the traditional modular polynomial, and we need to find the coefficients of $\mathcal{E}^*$, together with the isogeny. The idea of Elkies is to consider the same problems on the Tate curves associated to the elliptic curves $\mathcal{E}$ and $\mathcal{E}^*$.

To be brief, $\mathcal{E}$ has an equation in some parameter q, and the isogenous $\mathcal{E}^*$ is associated to parameter q^ℓ, where ℓ is the degree of the isogeny, which in our case is associated with a finite subgroup F of cardinality ℓ. To be more precise, we consider $\mathcal{E}$ has having equation $Y^2 = X^3 + AX + B$ with

$$A = -3E_4(q), B = -2E_6(q).$$

With a compatible scaling, we get the equation for $\mathcal{E}^* : Y^2 = X^3 + A^*X + B^*$ with

$$A^* = -3\ell^4 E_4(q^\ell),\ B^* = -2\ell^6 E_6(q^\ell).$$

More importantly, writing σ_r for the power sums of the roots of the kernel polynomial, we have

$$\sigma_1 = \frac{\ell}{2}(\ell E_2(q^\ell) - E_2(q)) = -\frac{\ell}{2}\mathcal{F}_\ell(q).$$

Beyond this, Elkies proved [20, formulas (66) to (69)]

Proposition 4.1.

$$A - A^* = 5(6\sigma_2 + 2A\sigma_0),$$
$$B - B^* = 7(10\sigma_3 + 6A\sigma_1 + 4B\sigma_0),$$

together with an induction relation satisfied by other σ_k for $k > 3$.

This can rephrased as (σ_1, A^*, B^*) is enough to describe an isogeny. Also A^* and B^* belong to $\mathbb{Q}[\sigma_1, A, B]$ since σ_2 and σ_3 do. The minimal polynomial of σ_1 is a modular polynomial, and we can express A^* and B^* as elements in the field $\mathbb{Q}[\sigma_1, A, B]$, which we use below. Rephrased another times, $E_4(q^\ell)$ and $E_6(q^\ell)$ are modular forms we need to express as known modular forms. See [20] for more details on this subject.

Given these quantities, there are several algorithms to get the isogeny. We refer to [8] for this.

4.3. **The CCR polynomials.**

4.3.1. *Reinterpreting Elkies's results.* One way of looking at the work of Elkies (taken from [12]) is to realize that we try to decompose the polynomial f_ℓ (say ℓ is odd) of degree $(\ell^2-1)/2$ over a subfield of degree $\ell+1$.

$$\begin{array}{c} \mathbb{Q}(A,B)[X]/(f_\ell(X,A,B)) \\ | \, (\ell-1)/2 \\ \mathbb{Q}(A,B)[X]/(U_\ell(X,A,B)) \\ | \, \ell+1 \\ \mathbb{Q}(A,B) \end{array}$$

A classical way for doing this is to use the trace t_1 of an element in $\mathbb{Q}(A,B)[X]/(f_\ell(X,A,B))$. Let x_1 stand for the (formal) abscissa of an ℓ-division point $P = (x_1, y_1) \neq O_E$. Other points are $P_j = [j]P = (x_j, y_j)$ and x_j can be expressed using division polynomials. For $0 \leq k \leq \ell+1$, we define

$$t_k = \sum_{j=1}^{d} x_j^k = \sum_{j=1}^{d} \left(x_1 - \frac{\psi_{j-1}(x_1)\psi_{j+1}(x_1)}{\psi_j(x_1)^2} \right)^k \tag{2}$$

so that $t_1 = x_1 + \cdots + x_d$ and $t_0 = d = (\ell-1)/2$. The minimal polynomial $U_\ell(X) = X^{\ell+1} + u_1 X^\ell + \cdots + u_0$ of t_1 defines the lower extension. Given the power sums t_k's, Newton's identities enable us to reconstruct the minimal polynomial $\prod_{i=1}^{d}(X - x_i) = X^d - t_1 X^{d-1} + \cdots$ over the intermediate extension. Putting everything together, t_1 coincides with σ_1 included above.

By direct application of (2), we can compute $U_3(X,Y,Z) = X^4 + 2YX^2 + 4ZX - Y^2/3$ (this is ψ_3 in disguised form). Larger values of ℓ require more work.

4.3.2. *Theory.* We start from an elliptic curve $\mathcal{E} : Y^2 = X^3 + AX + B$ and we fix some odd prime ℓ, putting $d = (\ell-1)/2$. Our aim is to find the equation of an ℓ-isogenous curve $\mathcal{E}^* : Y^2 = X^3 + A^*X + B^*$.

Theorem 4.2. *There exist three polynomials U_ℓ, V_ℓ, W_ℓ in $\mathbb{Z}[X,Y,Z,1/\ell]$ of degree $\ell+1$ in X such that $U_\ell(\sigma_1, A, B) = 0$, respectively $V_\ell(A^*, A, B) = 0$, $W_\ell(B^*, A, B) = 0$.*

Let us turn our attention to the properties of these polynomials.

Theorem 4.3. *When $\ell > 3$, the polynomials U_ℓ, V_ℓ, W_ℓ live in $\mathbb{Z}[X,Y,Z]$.*

Proposition 4.4. *Assigning respective weights 1, 2, 3 to X, Y, Z, the polynomials U_ℓ, V_ℓ and W_ℓ are homogeneous with weight $\ell+1$.*

Proposition 4.5. *The roots of $U_\ell(X, A(q), B(q))$ are $-\ell\mathcal{F}_\ell(q)/2$ and $\mathcal{F}_\ell(w\zeta_\ell^k)/2$ for $0 \le k < \ell$, where $w^\ell = q$ and ζ_ℓ is a root of unity.*

We follow [12, §8].

Proposition 4.6. *The height of U_ℓ (resp. V_ℓ, W_ℓ) is approximately $2k(\ell+1)\log\ell$ for $k = 1, 2, 3$ corresponding to U, V, W respectively.*

Proof: let $\lambda_k(u) = \sum_{n=1}^\infty \delta_{2k-1}(n)u^n$ which the important term in the sums. Using

$$\delta_{2k-1}(n) \le n^{2k-1}\sum_{d|n}\frac{1}{d^{2k-1}} \le \zeta(2k-1)n^{2k-1},$$

we can see that $\lambda_k(u)$ is dominated by $\zeta(2k-1)\sum_{n=1}^\infty n^{2k-1}u^n$ dominated by

$$(2k-1)!\,\zeta(2k-1)\sum_{n=1}^\infty \binom{n+2k-2}{2k-1}u^n = C_k u(1-u)^{-2k}.$$

So λ_k^m is dominated by $C_k^m(1-u)^{-2km}$, and the highest term is approximately $u^{m^2/2}$. An approximation to the largest coefficient of λ^m is

$$C_k^m\binom{m^2/2+2km-1}{2km-1} \approx C_k^m\frac{(m^2/2+2km)^{2km}}{(2km/e)}$$

$$= C_k^m\left(\frac{e}{4k}\right)^{2km}(m+4k)^{2km}.$$

Taking $m = \ell+1$ leads to the result. □

The traditional modular polynomial Φ_ℓ^t has height $6(\ell+1)\log\ell$ approximately [15, 39]. We see that U_ℓ and V_ℓ are smaller, but that W_ℓ is as big as Φ_ℓ^t. See the appendix for some tables.

4.3.3. *Computing isogenous curves over finite fields.* When using U_ℓ, V_ℓ, W_ℓ, we need to find the roots of three polynomials of degree $\ell+1$ instead of 1 in the traditional case. In general, if U_ℓ has rational roots (it should be 1, 2 or $\ell+1$), then this is the case for each of V_ℓ, W_ℓ. For each triplet of solutions (σ_1, z_1, z_2) we need to test whether this leads to an isogeny or not. See techniques for this task in [8].

5. Computing CCR polynomials

Using the results of the preceding section, the polynomials can be written as

$$U_\ell(X,Y,Z) = X^{\ell+1} + \sum_{i_1+2i_2+3i_3=\ell+1} u_{i_1,i_2,i_3} X^{i_1} Y^{i_2} Z^{i_3},$$
$$V_\ell(X,Y,Z) = X^{\ell+1} + \sum_{2i_1+2i_2+3i_3=\ell+1} v_{i_1,i_2,i_3} X^{i_1} Y^{i_2} Z^{i_3},$$
$$W_\ell(X,Y,Z) = X^{\ell+1} + \sum_{3i_1+2i_2+3i_3=\ell+1} w_{i_1,i_2,i_3} X^{i_1} Y^{i_2} Z^{i_3}.$$

All the methods to be described can be applied to U_ℓ, V_ℓ and W_ℓ. To simplify the presentation, we assume from now on that $\ell > 3$ and concentrate on U_ℓ, indicating what has to be changed for V_ℓ (resp. W_ℓ). We rewrite

$$U_\ell(X,Y,Z) = X^{\ell+1} + \sum_{r=0}^{\ell} X^r \sum_{2i_2+3i_3=\ell+1-r} c_{r,i_2,i_3} Y^{i_2} Z^{i_3}.$$

From this, we can see that there are no possible terms for $r = \ell$.

We first count the number of monomials. We take the following from [16, p. 110].

Proposition 5.1. *The number of solutions $\mathbf{N}_{1,2,3}(n)$ in positive integers of $i_1 + 2i_2 + 3i_3 = n$ is the closest integer to $(n+3)^2/12$.*

Using the same method

Proposition 5.2. *The number of solutions $\mathbf{N}_{2,3}(n)$ in positive integers of $2i_2 + 3i_3 = n$ is*

$$\mathbf{N}_{2,3}(n) = \frac{n+1}{6} + \frac{(-1)^n}{4} - \frac{1}{12} + \begin{cases} \frac{2}{3} & \text{if } n \equiv 0 \bmod 3, \\ \frac{1}{3} & \text{if } n \equiv 2 \bmod 3. \end{cases}$$

We are going to compute many products of the form $Y^{i_2} Z^{i_3}$ for Y and Z that are large floating point numbers or series. We use Pippenger's algorithm [34, pp. 247–249] (thanks to [5]) for that task.

The authors of [12] give three methods to compute the polynomial U_ℓ. The first is based on equation (2) and can be used for very small ℓ's. Two more methods use manipulations of q-expansions of series over $\mathbb{Q}$, or modulo small primes followed by recovery using the Chinese remaindering theorem using the bounds in Proposition 4.6.

5.1. **Using q-expansions.** Note that

$$\sigma_1(q) = -\frac{\ell \mathcal{F}_\ell(q)}{2} = \frac{\ell(\ell-1)}{2} + 12\ell \sum_{n=1}^{\infty} \sigma_1'(n) q^n$$

where $\sigma_1'(n)$ is the sum of the divisors of n prime to ℓ.

The second method proceeds by plugging the series $\sigma_1(q)$, $A(q)$ and $B(q)$ up to degree $\mathbf{N}_{1,2,3}(\ell+1) = O(\ell^2)$ (Proposition 5.1) in $U_\ell(X, Y, Z) = 0$ and find the unknown coefficients of the polynomial using a linear system over the rationals (in fact integers for $\ell > 3$). This can be done for small ℓ's using any mathematical system. Note that the number of algebraic operations (multiplications in $\mathbb{Q}$, or a finite field) will be close to $O(\ell^{2\omega})$ where ω is the constant for matrix multiplication with $2 \leq \omega \leq 3$, so typically $O(\ell^6)$, which is large.

The third method exploits the fact that the power sums $\sigma_r(q)$ for $1 \leq r \leq \ell+1$ are modular forms and can be represented as polynomials in $A(q)$ and $B(q)$ (or $E_4(q)$, $E_6(q)$)

$$\sigma_r(q) = \sum_{2i_2+3i_3=r} u_{r,i_2,i_3} A(q)^{i_2} B(q)^{i_3}.$$

This leads to a linear system $\mathcal{S}_r$ (independent of ℓ) in the u_{r,i_2,i_3}'s. By Proposition 5.2, the system has size $\mathbf{N}_{2,3}(r) \times \mathbf{N}_{2,3}(r) \approx (r/6)^2$ and the linear system can be solved with $O(r^\omega)$ operations over $\mathbb{Z}$, for a total of $O(\sum_r r^\omega) = O(\ell^{\omega+1}) = O(\ell^4)$ generally. Note that all these systems may be solved in parallel.

The authors of [12] suggest to use a more adapted basis. When $r = 2m$, use $\{E_4^{m-3j}\Delta^k, 0 \leq k \leq \lfloor m/3 \rfloor\}$; when $r = 2m+3$ (remember that the coefficient for $r = 1$ is 0), use $\{E_6 E_4^{m-3k}\Delta^k, 0 \leq k \leq \lfloor m/3 \rfloor\}$. Note that in all cases, the series for index k start with q^k and the bases present a triangular shape, in other words the corresponding system $\mathcal{S}_r$ is triangular. Moreover, since the leading coefficient is 1, all solutions to the system will be integers.

Once solved for all r's, we use Newton's identities to recover the coefficients of U_ℓ. One needs to evaluate the series $\sigma_r(q)$ using intermediate expressions in $w = q^{1/\ell}$ having roots of unity ζ_ℓ temporarily appearing and vanishing. See [21, §2.2] for more details and complexity analysis. In particular, if we denote by $\mathsf{M}_q(d)$ the number of arithmetic operations in $\mathbb{Z}$ required to multiply two dense q-expansions with d terms, then the total complexity of the series computations is $O(\ell \mathsf{M}_q(\ell d)$, which is $O(\ell \mathsf{M}_q(\ell^2))$ in our case. If H is a bound on the height of the polynomial, then the bit complexity is $O(\ell^3(\log \ell)\mathsf{M}(H))$. Assuming $H \in O(\ell \log \ell)$ by Proposition 4.6, this is $O(\ell^4 \log^{3+\epsilon} \ell)$.

Example. Consider the case $\ell = 5$. The systems $\mathcal{S}_r$ to be solved come from the equations:

$$\begin{aligned}
\sigma_2(q) &= u_{2,0}E_4 \\
\sigma_3(q) &= u_{3,0}E_6 \\
\sigma_4(q) &= u_{4,0}E_4^2 \\
\sigma_5(q) &= u_{5,0}E_6E_4 \\
\sigma_6(q) &= u_{6,0}E_4^3 + u_{6,1}\Delta
\end{aligned}$$

We compute

$$\sigma_6(q) = 1000320 + 186071040q + \cdots$$

and we remember that $E_4(q) = 1 + 240q + \cdots$, $\Delta(q) = q + \cdots$ so that the system $\mathcal{S}_6$ is

$$\left\{ \begin{array}{rcl} 1000320 &=& u_{6,0} \\ 186071040 &=& 720\, u_{6,0} + u_{6,1} \end{array} \right.$$

which is triangular indeed and therefore easy to solve. Its solutions are integers.

We can also work modulo small primes and use the Chinese Remainder Theorem to recover the polynomials.

5.2. **Floating point methods.** We adapt the methods proposed for ordinary modular equations to our triplet of polynomials (U_ℓ, V_ℓ, W_ℓ). We note H for the logarithmic height of the polynomials, that we have estimated to $2k(\ell+1)\log \ell$ in Proposition 4.6.

5.2.1. *Solving a linear system.* We start from

$$U_\ell(\sigma_1, A, B) = 0 = \sigma_1(q)^{\ell+1}$$

$$+ \sum_{r=0}^{\ell} \sigma_1(q)^r \sum_{2i_2+3i_3=\ell+1-r} c_{r,i_2,i_3} A^{i_2}(q) B^{i_3}(q)$$

and we compute floating point values to get a linear system in the c_{r,i_2,i_3} that should come out as integers for $\ell > 3$. We evaluate $\sigma_1(q)$, $A(q) = -3E_4(q)$ and $B(q) = -2E_6(q)$ at high precision for chosen values of τ in $q = \exp(2i\pi\tau)$. This would involve $O(\ell^{2\omega})$ floating point operations, and we can do better in the next section.

5.2.2. *Using power sums.* The case of the traditional modular polynomial is treated in [21]. We can use the same approach for our polynomials. First of all, we need to compute

$$-\frac{\ell}{2}\mathcal{F}_\ell(q), \{\mathcal{F}_\ell(w\zeta_\ell^k)/2, 0 \le k < \ell\}.$$

By definition

$$\mathcal{F}_\ell(q) = E_2(q) - \ell E_2(q^\ell)$$

and

$$\mathcal{F}_\ell(w\zeta_\ell^k) = E_2(w\zeta_\ell^k) - \ell E_2(q),$$

and the last term is a constant w.r.t. k. We first evaluate $E_2(q)$, $E_2(q^\ell)$ and then the other roots, by sharing the computations: All terms we need are of the form $(w\zeta_\ell^k)^e = w^e\zeta_\ell^{(ke) \bmod \ell}$. When $\ell \mid e$, the computation is a little faster. We give the corresponding code as Algorithm 3. We also precompute $\xi_k = \zeta_\ell^k$. The complete code is in Algorithm 4.

Algorithm 3: Combined evaluation of $T_{2k}(w\zeta_\ell^j)$.

Function *EvaluateConjugateValues(ℓ, w, N, (ξ), $kmax$)*
- **Input :** ℓ, w, (ξ), N
- **Output:** $(T_{2k}(w\zeta_\ell^j))$ for $0 \le k \le kmax$, $0 \le j < \ell$
- 1. **for** $k := 0$ **to** $kmax$ **do**
  - **for** $j := 0$ **to** $\ell - 1$ **do**
    - $T[k, j] \leftarrow 0$
- 2. $s \leftarrow 1$; $A \leftarrow \{1\}$; $W[1] \leftarrow \{w\}$; $c \leftarrow 0$
- 3. **for** $n := 1$ *while* $n(3n + 1)/2 \le N$ **do**
  - $s \leftarrow -s$
  - // $s = (-1)^n$
  - 3.1 $c \leftarrow c + 2n - 1$
  - 3.2 **for** $r := 1$ **to** 2 **do**
    - 3.2.1 **if** $r = 2$ **then**
      - $c \leftarrow c + n$
      - // $c = n(3n + 1)/2$
    - 3.2.2 $w' \leftarrow s\cdot$FindPowerInTable(A, W, c)
    - 3.2.3 $C \leftarrow (6n + (-1)^r)^2$
    - 3.2.4 **for** $k := 0$ **to** $kmax$ **do**
      - $T[k, 0] \leftarrow T[k, 0] + w'$
      - **for** $j := 1$ **to** $\ell - 1$ **do**
        - $T[k, j] \leftarrow T[k, j] + \xi[(jc) \bmod \ell]\, w'$
      - **if** $k < kmax$ **then**
        - $w' \leftarrow C \cdot w'$
- 4. **for** $k := 0$ **to** $kmax$ **do**
  - **for** $j := 0$ **to** $\ell - 1$ **do**
    - $T[k, j] \leftarrow T[k, j] + 1$
- 5. **return** T.

The system $\mathcal{S}_r$ has size $N_{2,3}(r)^2 = O(r^2)$ and we need $O(r^\omega)$ operations to solve it, for a total of $O(\ell^{\omega+1})$. Like in the series case, we can use the (E_4, E_6, Δ) basis since we anticipate integer coefficients, as indicated by the series computations.

Algorithm 4: Computing CCR polynomial k using floating point numbers.

Function *ComputeUVW(k, ℓ)*
- **Input :** $k \in \{1,2,3\}$ corresponding to U_ℓ, V_ℓ or W_ℓ respectively, ℓ an odd prime
- **Output:** the CCR polynomial of order k
- 0.0 $H \leftarrow 2k(\ell+1)\log \ell$; all computations are carried out at precision H
- 0.1 compute $\zeta_\ell \leftarrow \exp(2i\pi/\ell)$
- 0.2 **for** $j := 0$ **to** $\ell - 1$ **do**
  - $\xi_j \leftarrow \zeta_\ell^j$
- 0.3 Compute all systems $\mathcal{S}_r$ for $2 \le r \le \ell + 1$
- 0.4 **for** $r := 2$ **to** $\ell + 1$ **do**
  - $\mathcal{L}_r \leftarrow \emptyset$
- 0.5 $\rho \leftarrow 1$
- 1. **while** *there is a system $\mathcal{L}_r$ that is not solved* **do**
  - 1.0 $\rho \leftarrow \rho + 0.1$
  - 1.1 $w \leftarrow \exp(-2\pi\rho/\ell)$
  - 1.2 $T \leftarrow$ EvaluateConjugateValues(ℓ, w, N, (ξ), $2k$)
  - 1.3. use Theorem 3.1 to evaluate E_{2k} for all q's from T, yielding (γ_r) for $r = 0, \ldots, \ell + 1$
  - 1.4 **for** $r := 2$ **to** $\ell + 1$ **do**
    - **if** $\mathcal{L}_r$ *is not solved* **then**
      - 1.4.1 Instantiate $\mathcal{S}_r$ with $\sum \gamma_i^r$, A_ρ and B_ρ; add it to $\mathcal{L}_r$
      - 1.4.2 **if** $\mathcal{L}_r$ *has as many equations as unknowns* **then**
        - solve $\mathcal{L}_r$ and store the values; declare $\mathcal{L}_r$ solved
- 2. Round the coefficients and use Newton's formulas.

Example. Take again $\ell = 5$, for which the $\mathcal{S}_r$ were already given. Let us concentrate on the case of $\sigma_6(q) = u_{6,0}E_4^3 + u_{6,1}\Delta$; we start with $\mathcal{L}_6 = \emptyset$. Using $\rho = 1.1$ leads to

$$\mathcal{L}_6 = \{1.912407642u_{6,0} + 0.0009726854527956u_{6,1}$$
$$= 1393450.57337539139\}$$

and the following iteration with $\rho = 1.2$ adds

$$\{1.435895343u_{6,0} + 0.0005247501300701u_{6,1}$$
$$= 1156054.63606077432\}$$

and the solution of $\mathcal{L}_6$ (rounded to integers) is

$$u_{6,1} = -534159360, u_{6,0} = 1000320.$$

5.3. **Isogeny volcanoes.** The method in [13] shares many common points with the method to be described next but with a worse complexity. It uses supersingular curves whose complete explicit ℓ-torsion is required.

The work of [10] is a building block in [39] where direct evaluation of $\Phi_\ell(X, j(E)) \bmod q$ is made possible using an explicit version of the Chinese remainder theorem modulo small primes. Our version leads an easy adaptation to this problem for the CRT polynomials.

5.3.1. *Quick presentation.* In a nutshell, the algorithm in [10] performs computations modulo special primes p satisfying arithmetical conditions: $p \equiv 1 \bmod \ell$ and $4p = t^2 - \ell^2 v^2 D$ in integers t and v, v not a multiple of ℓ; $D < 0$ is the discriminant of some auxiliary quadratic field. With these conditions, the so-called class polynomial $H_D(X)$ splits completely modulo p and its roots are j-invariants of elliptic curves with complex multiplication that are needed in the algorithms. Basically, the algorithm interpolates data using the roots of $H_D(X)$. The central point is to compute isogenies between the two levels of the isogeny volcano associated to D modulo p. We refer the reader to the original article for more properties related to elliptic curves. For our purpose, we just need to know that we have isogeny data available and that they can help us computing the polynomial $U_\ell(X, A, B) \bmod p$ from these data. We refer to the article for the complexity under GRH, namely $O(\ell^3 (\log \ell)^3 \log\log \ell)$ using $O(\ell^2 \log(\ell p))$ space for suitably chosen p.

5.3.2. *The algorithm.* In our case, we consider power sums again. To lighten the exposition, we consider U_ℓ only since the algorithm is the same for V_ℓ and W_ℓ.

We adapt here a slight modification of the simplified version Algorithm 2.1 of [10] to our needs. All we describe is also valid in the full version in [10]. We denote by $\mathcal{P}$ the powersums of U_ℓ viewed as a polynomial in X with coefficients in $\mathbb{Z}[Y, Z]$ for $1 \leq r \leq \ell + 1$. We can write

$$\mathcal{P}_r(Y, Z) = \sum_{2i_2 + 3i_3 = r} c_{r,i_2,i_3} Y^{i_2} Z^{i_3}.$$

These sums will be reconstructed from values $\mathcal{P}_{r,i}(A_i, B_i)$ associated to curves $\mathcal{E}_i : Y^2 = X^3 + A_i X + B_i$.

Step 3 is done as follows. Select a random point P of order ℓ on $E_i/\mathbb{F}_p = [A_i, B_i]$. Compute the rational isogeny $\mathcal{E}_i \to \mathcal{E}_i' = \mathcal{E}_i/\langle P \rangle$ using Vélu's formulas. If $j(\mathcal{E}_i')$ is a root of H_D, then $\mathcal{E}_i'$ is on the crater and is one of the two neighbours. If it does not belong to the crater, it belongs to the floor. Identification details in the general case with multiple volcanoes is treated in [10].

For V_ℓ (resp. W_ℓ), replace σ_1 by A^* (resp. B^*) in Step 5 as far as reconstruction is concerned.

Algorithm 5: Computing $U_\ell(X, Y, Z) \bmod p$

Function ComputeUMod(ℓ, D, $H_D(X)$, p):

Input : ℓ an odd prime, D the discriminant of an imaginary quadratic order $\mathcal{O}$ of discriminant D with class number $h(D) \geq \ell + 2$; H_D is the class polynomials associated to order $\mathcal{O}$; p prime with $p \equiv 1 \bmod \ell$ and $4p = t^2 - \ell^2 v^2 D$, $v \not\equiv 0 \bmod \ell$

Output: $U_\ell(X, Y, Z) \bmod p$

1. Build the list $\mathcal{J}_D$ containing the roots of $H_D(X)$ modulo p
2. For each $j_i \in \mathcal{J}_D$, find a curve $\mathcal{E}_i$ having invariant j_i and cardinality $m = p + 1 - t$; call $\mathcal{C}_D$ this set of curves
3. For each $\mathcal{E}_i \in \mathcal{C}_D$ find all its neighbors $\mathcal{N}(\mathcal{E}_i)$ in the volcano: 2 horizontal isogenies and $\ell - 1$ on the floor. This is a collection of $\ell + 1$ triplets (σ_1, A^*, B^*) obtained via Vélu's formulas
4. For each $\mathcal{E}_i : Y^2 = X^3 + A_i X + B_i$, compute the power sums $\mathcal{P}_{r,i}(A_i, B_i)$ corresponding to the values of σ_1 in $(\sigma_1, A^*, B^*) \in \mathcal{N}(\mathcal{E}_i)$ using Vélu's formulas
5. Reconstruct the powers sums $\mathcal{P}_r(Y, Z) \bmod p$ using A, B and the $\mathcal{P}_{r,i}(A_i, B_i)$, solving a $N_{2,3}(r) \times N_{2,3}(r)$ linear system
6. **return** $\mathcal{P}$.

5.3.3. *A numerical example.* Let us give one value for $\ell = 5$. We select $D = -71$ for which $h(-71) = 7 \geq 5 + 2$. Consider $p = 1811$. The roots of $H_{-71}(X)$ modulo p are:

$$\mathcal{J}_D = \{313, 1073, 1288, 1312, 1402, 1767, 1808\}.$$

Associated are curves and neighbours for each j value. These can be found in Table 1. The power sums $\mathcal{P}_r$ corresponding to the values are:

$\mathcal{E}_i \backslash r$	1	2	3	4	5	6
$[1582, 902]$	0	105	1680	1379	756	772
$[1662, 405]$	0	527	1188	90	748	888
$[1451, 1331]$	0	1723	403	350	293	583
$[1013, 747]$	0	1133	18	1594	1738	105
$[224, 753]$	0	95	760	1790	1603	27
$[1128, 1504]$	0	155	669	1424	1130	522
$[91, 725]$	0	1793	1523	589	1233	134

For instance, $\sigma_6 = c_{030} Y^3 + c_{012} Z^2$, we need to solve

$$\begin{cases} 772 &= c_{030} 1582^3 + c_{012} 902^2 \bmod 1811, \\ 888 &= c_{030} 1662^3 + c_{012} 405^2 \bmod 1811, \\ 583 &= c_{030} 1451^3 + c_{012} 1331^2 \bmod 1811, \\ \cdots & \cdots \quad \cdots \end{cases}$$

that is $1565Y^3 + 1218Z^2$. The coefficients are:

$$\begin{aligned}\sigma_2 &= 1771Y\\ \sigma_3 &= 1331Z\\ \sigma_4 &= 1120Y^2\\ \sigma_5 &= 341YZ\\ \sigma_6 &= 1565Y^3 + 1218Z^2\end{aligned}$$

6. Computing A^* and B^* as rational fractions

From [33, Theorem 3.9], there exist polynomials $N_{\ell,A}$ and $N_{\ell,B}$ of degree less than $\ell+1$ such that

$$A^* = \frac{N_{\ell,A}(X,A,B)}{U'_\ell(X)},\ B^* = \frac{N_{\ell,B}(X,A,B)}{U'_\ell(X)}$$

(Only here: $U'_\ell(X) = \frac{\partial U_\ell}{\partial X}$.) Moreover, $N_{\ell,A}$ (resp. $N_{\ell,B}$) are polynomials with integer coefficients and of generalized weight $2\ell+4$ (resp. $2\ell+6$). The authors of the reference use Groebner basis computations to find the two numerators.

Alternatively, given U_ℓ, we can start from (the same is true for $N_{\ell,B}$): $U'_\ell(X,A,B)A^* = N_{\ell,A}(X,A,B)$ and we plug the series to get

$$U'_\ell(\sigma_1(q),A(q),B(q))(-\ell^4 E_4(q^\ell)) = N_{\ell,A}(\sigma_1(q),A(q),B(q)).$$

We find the coefficients by solving a linear system (over $\mathbb{Q}$ or using small primes as already described). We can precompute the powers of the series for σ_1, A and B and remark that U'_ℓ and $N_{\ell,A}$ share a lot of them. Also, the series $E_4(q^\ell)$ is rather sparse, so that the product with this quantity is fast.

We can also use floating point numbers as above and recognize integers in the values of the coefficients of $N_{\ell,A}$.

There is an advantage to compute $N_{\ell,A}$ and $N_{\ell,B}$ at the same time, sharing as many powers as possible, all the more in our use of Pippenger's algorithm. Numerical examples are given in the appendix.

7. Implementation and numerical results

A lot of trials were done using Maple programs, some of which were then rewritten in Magma (version 2.26-10), for speed. See the author's web page. Computing the polynomials for $\ell \leq 100$ takes a few minutes on a classical laptop. Checking them is done using SEA, as mentioned in [33].

We give some examples of the *relative height* $\tilde{H}$ for some of our polynomials. Here $\tilde{H}(P) = H(P)/((\ell+1)\log\ell)$. Note that these quantities seem to stabilize when ℓ increases.

ℓ	$\tilde{H}(\Phi_\ell^t)$	$\tilde{H}(\Phi_\ell^c)$	$\tilde{H}(\Phi_\ell^*)$	$\tilde{H}(U_\ell)$	$\tilde{H}(U_\ell^*)$
5	11.243	0.762	$--$	0.526	$--$
7	9.787	0.582	$--$	0.640	$--$
11	10.130	1.842	1.120	0.670	0.240
13	9.565	0.367	0.941	0.688	$--$
17	9.581	0.958	0.714	0.690	$--$
19	9.365	0.648	0.630	0.695	$--$
23	9.438	1.995	0.419	0.698	0.441
101	$--$	1.111	0.159	0.778	$--$
103	$--$	0.740	0.249	0.779	$--$
107	$--$	2.218	0.228	0.781	0.493
109	$--$	0.379	0.213	0.782	$--$

Data are computed using the polynomials available in MAGMA: Φ_ℓ^c is called *canonical polynomial* and Φ_ℓ^* is called *Atkin polynomial*. In the case of Φ_ℓ^c, the height depends on $\ell \bmod 12$. See the appendix for more statistics on the sizes of these polynomials.

When $\ell \equiv 11 \bmod 12$, Atkin [1] suggests to replace σ with $f(q) = (\eta(q)\eta(q^\ell))^2$. For instance:

$$U_{11}^*(X) = X^{12} - 990\,\Delta\,X^6 + 440\,\Delta\,E_4\,X^4 - 165\,\Delta\,E_6\,X^3 + 22\,\Delta\,E_4^2 X^2 - \Delta\,E_4\,E_6\,X - 11\,\Delta^2,$$

whose height is smaller than that of the other alternatives. Using this type of equation for computing isogenies requires more work, see [32].

8. Conclusions

We have given a lot of methods for computing the Charlap-Coley-Robbins polynomials, including representations as fractions in polynomials. In isogeny cryptography they are useful for relatively small ℓ's, if we store $(U_\ell, N_{\ell,A}, N_{\ell,B})$. If one wants to compute an isogeny, it is enough to compute a root of U_ℓ followed by instantiations of three polynomials.

There is an alternative to this, suggested by Atkin [1], using partial derivatives of U_ℓ. This is described in [32].

Acknowledgments. The author wishes to thank A. Bostan and F. Chyzak for helpful discussions around some aspects of this work; special thanks to the former for his impressive list of references for the fast evaluation of hypergeometric functions. Thanks also to L. De Feo for his updates on cryptographic applications of isogenies.

References

[1] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime (II). Draft. Available on `http://listserv.nodak.edu/archives/nmbrthry.html`, 1992.

[2] B. C. Berndt. Ramanujan's formulas for Eisenstein series. In *Number theory and related topics (Bombay, 1988)*, volume 12 of *Tata Inst. Fund. Res. Stud. Math.*, pages 23–29. Tata Inst. Fund. Res., Bombay, 1989.
[3] B. C. Berndt, H. H. Chan, J. Sohn, and S. H. Son. Eisenstein series in Ramanujan's lost notebook. *Ramanujan J.*, 4(1):81–114, 2000.
[4] B. C. Berndt and A. J. Yee. Ramanujan's contributions to Eisenstein series, especially in his lost notebook. In *Number theoretic methods (Iizuka, 2001)*, volume 8 of *Dev. Math.*, pages 31–53. Kluwer Acad. Publ., Dordrecht, 2002.
[5] D. J. Bernstein. Pippenger's exponentiation algorithm. `https://cr.yp.to/papers.html`, January 2002.
[6] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, volume 265 of *London Math. Soc. Lecture Note Ser.* Cambridge University Press, 1999.
[7] J. M. Borwein and P. B. Borwein. A cubic counterpart of Jacobi's identity and the AGM. *Trans. Amer. Math. Soc.*, 323(2):691–701, 1991.
[8] A. Bostan, F. Morain, B. Salvy, and É. Schost. Fast algorithms for computing isogenies between elliptic curves. *Math. Comp.*, 77(263):1755–1778, 2008.
[9] R. Brent and P. Zimmermann. *Modern Computer Arithmetic*. Cambridge University Press, 2010.
[10] R. Bröker, K. E. Lauter, and A. V. Sutherland. Modular polynomials via isogeny volcanoes. *Math. Comput.*, 81(278):1201–1231, 2012.
[11] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: an efficient post-quantum commutative group action. In T. Peyrin and S. D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.
[12] L. S. Charlap, R. Coley, and D. P. Robbins. Enumeration of rational points on elliptic curves over finite fields. Draft, 1991.
[13] D. Charles and K. Lauter. Computing modular polynomials. *LMS J. Comput. Math.*, 8:195–204, 2005.
[14] D. X. Charles, K. E. Lauter, and E. Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptol.*, 22(1):93–113, 2009.
[15] P. Cohen. On the coefficients of the transformation polynomials for the elliptic modular function. *Math. Proc. Cambridge Philos. Soc.*, 95:389–402, 1984.
[16] L. Comtet. *Advanced Combinatorics*. D. Reidel Publishing Company, 1974.
[17] J.-M. Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. `http://eprint.iacr.org/2006/291`.
[18] R. Dupont. Fast evaluation of modular functions using Newton iterations and the AGM. *Math. Comp.*, 80(275):1823–1847, 2011.
[19] A. El Basraoui and A. Sebbar. Zeros of the Eisenstein series E_2. *Proc. Amer. Math. Soc.*, 138(7):2289–2299, 2010.
[20] N. D. Elkies. Elliptic and modular curves over finite fields and related computational issues. In D. A. Buell and J. T. Teitelbaum, editors, *Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A. O. L. Atkin*, volume 7 of *AMS/IP Studies in Advanced Mathematics*, pages 21–76. American Mathematical Society, International Press, 1998.
[21] A. Enge. Computing modular polynomials in quasi-linear time. *Math. Comp.*, 78(267):1809–1824, 2009.
[22] A. Enge, W. Hart, and F. Johansson. Short addition sequences for theta functions. *J. Integer Seq.*, 21(2):Art. 18.2.4, 34, 2018.
[23] A. Erdélyi, editor. *Higher transcendental functions*, volume II. McGraw-Hill, 1953.
[24] L. D. Feo, J. Kieffer, and B. Smith. Towards practical key exchange from ordinary isogeny graphs. In T. Peyrin and S. D. Galbraith, editors, *Advances in Cryptology -*

ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III, volume 11274 of *Lecture Notes in Computer Science*, pages 365–394. Springer, 2018.

[25] L. D. Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski. Sqisign: Compact post-quantum signatures from quaternions and isogenies. In S. Moriai and H. Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2020.

[26] D. Jao and L. De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In B. Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011.

[27] F. Johansson. Computing hypergeometric functions rigorously. *ACM Trans. Math. Softw.*, 45(3):30, 2019.

[28] M. Kaneko and D. Zagier. Supersingular j-invariants, hypergeometric series, and Atkin's orthogonal polynomials. In D. A. Buell and J. T. Teitelbaum, editors, *Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A. O. L. Atkin*, volume 7 of *AMS/IP Studies in Advanced Mathematics*, pages 97–126. American Mathematical Society, International Press, 1998.

[29] M. Kaneko and M. Koike. On modular forms arising from a differential equation of hypergeometric type. *Ramanujan J.*, 7(1-3):145–164, 2003. Rankin memorial issues.

[30] H. Labrande. Computing Jacobi's theta in quasi-linear time. *Math. Comp.*, 87(311):1479–1508, 2018.

[31] M. Mezzarobba and B. Salvy. Effective bounds for p-recursive sequences. *J. Symb. Comput.*, 45(10):1075–1096, 2010.

[32] F. Morain. Using the Charlap-Coley-Robbins polynomials for computing isogenies between elliptic curves. In preparation, February 2023.

[33] M. Noro, M. Yasuda, and K. Yokoyama. Symbolic computation of isogenies of elliptic curves by Vélu's formula. *Comment. Math. Univ. St. Pauli*, 68:93–130, 2020.

[34] N. Pippenger. On the evaluation of powers and monomials. *SIAM J. Comput.*, 9(2):230–250, 1980.

[35] A. Poteaux and É. Schost. Modular composition modulo triangular sets and applications. *Comput. Complexity*, 22(3):463–516, 2013.

[36] R. A. Rankin. *Modular forms and functions*. Cambridge University Press, 1977.

[37] A. Rostovtsev and A. Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. `http://eprint.iacr.org/`.

[38] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7:219–254, 1995.

[39] A. V. Sutherland. On the evaluation of modular polynomials. In *ANTS X—Proceedings of the Tenth Algorithmic Number Theory Symposium*, volume 1 of *Open Book Ser.*, pages 531–555. Math. Sci. Publ., Berkeley, CA, 2013.

[40] J. van der Hoeven. Fast evaluation of holonomic functions. *Theor. Comput. Sci.*, 210(1):199–215, 1999.

[41] J. van der Hoeven. Fast evaluation of holonomic functions near and in regular singularities. *J. Symb. Comput.*, 31(6):717–743, 2001.

Appendix A. Some values of U_ℓ, etc.

Note there is a sign flip compared to [33] due to a normalization different from the reference, but used in other articles.

For $\ell = 3$, we compute

$$V_3(X, A, B) = X^4 - 84AX^3 + 246A^2X^2 + (63756A^3 + 432000B^2)X$$
$$+576081A^4 + 3888000AB^2,$$
$$W_3(X, A, B) = X^4 + 732BX^3 + (25088A^3 + 171534B^2)X^2$$
$$+(1630720A^3B + 11009548B^3)X - 139150592A^3B^2 - 437245479B^4$$
$$-\frac{297493504}{27}A^6;$$
$$N_{3,A}(X, A, B) = 84X^3A - 360X^2B - 76XA^2 + 36AB,$$
$$N_{3,B}(X, A, B) = 732X^3B + \frac{1456}{3}X^2A^2 - 724XAB - \frac{112}{3}A^3 + 108B^2.$$
$$N_{5,A} = 630AX^5 - 9360BX^4 - 8240A^2X^3 + 24480BAX^2$$
$$+(1120A^3 - 28800B^2)X - 3200BA^2,$$
$$N_{5,B} = 15630\,X^5B + 34720\,X^4A^2 - 208240\,X^3AB$$
$$+\left(-76160\,A^3 + 110400\,B^2\right)X^2 + 138720\,XA^2B - 83200\,AB^2.$$

For P a polynomial with rational coefficients, we approximate its size by the sum S of the number of bits of the absolute value of each coefficient. For instance, we this criterion, we find that $S(U_5) = 36$. The total for $(U_5, N_{5,A}, N_{5,B})$ is $36 + 91 + 117 = 244$ compared to $S(V_5) = 349$, $S(W_5) = 602$; $S(\Phi_5^t) = 2838$, $S(\Phi_5^c) = 55$. We note $S_{tot}(\ell) = S(U_\ell) + S(N_{\ell,A}) + S(N_{\ell,B})$.

ℓ	$\tilde{H}(V_\ell)$	$\tilde{H}(W_\ell)$	$\tilde{H}(N_{\ell,A})$	$\tilde{H}(N_{\ell,B})$	$S_{tot}(\ell)$
5	3.266	4.336	1.063	1.268	244
7	3.050	4.207	0.973	1.167	551
11	2.939	3.979	0.896	1.016	1816
13	2.856	3.969	0.864	0.983	2684
17	2.770	3.883	0.831	0.919	5762
19	2.754	3.831	0.820	0.901	7866
23	2.723	3.764	0.799	0.869	13632
101	2.471	3.527	0.801	0.818	1187388
103	2.469	3.517	0.801	0.818	1262059
107	2.466	3.516	0.803	0.819	1422237
109	2.467	3.515	0.803	0.819	1504765

Appendix B. Numerical data for the isogeny volcano algorithm

LIX - Laboratoire d'informatique de l'École polytechnique *and* GRACE - Inria Saclay–Île-de-France

Email address: morain@lix.polytechnique.fr

i	$\mathcal{E}_i = [A_i, B_i]$	$\mathcal{E}_i^*$	$\sigma(\mathcal{E}_i^*)$
1	$[1582, 902]$	$[594, 422]$	226
		$[1543, 911]$	1542
		$[937, 1244]$	1283
		$[1333, 561]$	1691
		$[879, 342]$	1212
		$[757, 1578]$	1290
2	$[1662, 405]$	$[1770, 433]$	529
		$[1439, 1411]$	1536
		$[259, 355]$	1810
		$[382, 1793]$	1733
		$[1472, 543]$	433
		$[413, 1603]$	1203
3	$[1451, 1331]$	$[1096, 1433]$	743
		$[1371, 1367]$	98
		$[1105, 1195]$	207
		$[1657, 1699]$	787
		$[811, 812]$	1769
		$[779, 1311]$	18
4	$[1013, 747]$	$[1691, 473]$	1705
		$[509, 342]$	1245
		$[1642, 417]$	1406
		$[127, 765]$	1519
		$[905, 1464]$	145
		$[1277, 254]$	1224
5	$[224, 753]$	$[1485, 892]$	1566
		$[823, 1106]$	908
		$[397, 1451]$	1729
		$[131, 673]$	450
		$[654, 1798]$	1353
		$[1805, 1025]$	1238
6	$[1128, 1504]$	$[1275, 1672]$	1176
		$[1409, 761]$	1362
		$[907, 1757]$	309
		$[824, 1267]$	781
		$[578, 1320]$	1208
		$[1168, 1207]$	597
7	$[91, 725]$	$[1184, 542]$	1284
		$[1753, 297]$	859
		$[1440, 1524]$	1268
		$[421, 410]$	517
		$[1626, 1013]$	245
		$[198, 159]$	1260

TABLE 1. Values for $\ell = 5$ and $p = 1811$.