# On the evaluation of modular polynomials

Andrew V. Sutherland

Massachusetts Institute of Technology

ANTS X — July 10, 2012

`http://math.mit.edu:/~drew`

## Introduction

Let $\ell$ be a prime and let $\mathbb{F}_q$ be a finite field (assume $q$ is prime).

**Problem:** Given an elliptic curve $E/\mathbb{F}_q$, identify any and all curves $E'/\mathbb{F}_q$ that are $\ell$-isogenous to $E$.

This problem arises in many applications, and it is often the computationally dominant step.

## Introduction

Let $\ell$ be a prime and let $\mathbb{F}_q$ be a finite field (assume $q$ is prime).

**Problem:** Given an elliptic curve $E/\mathbb{F}_q$, identify any and all curves $E'/\mathbb{F}_q$ that are $\ell$-isogenous to $E$.

This problem arises in many applications, and it is often the computationally dominant step.

**Solution:** Compute the polynomial

$$\phi_\ell(Y) = \Phi_\ell(j(E), Y),$$

and find its roots in $\mathbb{F}_q$. Here $\Phi_\ell \in \mathbb{Z}[X, Y]$ is the classical modular polynomial that parameterizes pairs of $\ell$-isogenous elliptic curves.

If $\ell$ is at all large, say $\ell = \Omega(\log q)$, the hard part is computing $\phi_\ell$. Finding its roots is easy by comparison.

## The Modular Polynomial $\Phi_\ell(X,Y)$

$\Phi_\ell \in \mathbb{Z}[X,Y]$ is symmetric, with degree $\ell+1$ in both $X$ and $Y$.
Its total size is $O(\ell^3 \log \ell)$ bits.

| $\ell$ | coefficients | largest | average | total |
|---|---:|---:|---:|---:|
| 127 | 8258 | 7.5kb | 5.3kb | 5.5MB |
| 251 | 31880 | 16kb | 12kb | 48MB |
| 503 | 127262 | 36kb | 27kb | 431MB |
| 1009 | 510557 | 78kb | 60kb | 3.9GB |
| 2003 | 2009012 | 166kb | 132kb | 33GB |
| 3001 | 4507505 | 259kb | 208kb | 117GB |
| 4001 | 8010005 | 356kb | 287kb | 287GB |
| 5003 | 12522512 | 454kb | 369kb | 577GB |
| 10009 | 50085038 | 968kb | 774kb | 4.8TB |

**Size of** $\Phi_\ell(X,Y)$

## The Modular Polynomial $\Phi_\ell(X,Y)$

$\Phi_\ell \in \mathbb{Z}[X,Y]$ is symmetric, with degree $\ell+1$ in both $X$ and $Y$.
Its total size is $O(\ell^3 \log \ell)$ bits.

| $\ell$ | coefficients | largest | average | total |
|---|---:|---:|---:|---:|
| 127 | 8258 | 7.5kb | 5.3kb | 5.5MB |
| 251 | 31880 | 16kb | 12kb | 48MB |
| 503 | 127262 | 36kb | 27kb | 431MB |
| 1009 | 510557 | 78kb | 60kb | 3.9GB |
| 2003 | 2009012 | 166kb | 132kb | 33GB |
| 3001 | 4507505 | 259kb | 208kb | 117GB |
| 4001 | 8010005 | 356kb | 287kb | 287GB |
| 5003 | 12522512 | 454kb | 369kb | 577GB |
| 10009 | 50085038 | 968kb | 774kb | 4.8TB |

**Size of** $\Phi_\ell(X,Y)$

But for $\ell = 10009$ and $q \approx 2^{256}$, the size of $\phi_\ell$ is just 320KB!
Even with $\log q \approx \ell = 10009$ it is under 20MB.

## Point-counting at 2500 digits

*“. . . Despite this progress, computing modular polynomials remains the stumbling block for new point counting records. Clearly, to circumvent the memory problems, one would need an algorithm that directly obtains the polynomial specialised in one variable.”*

INRIA Project-Team TANC Report

This record was set in December 2006.

## Computing $\Phi_\ell$ with the CRT

Strategy: compute $\Phi_\ell \bmod p$ for sufficiently many primes $p$ and use the CRT to compute $\Phi_\ell$ (or $\Phi_\ell \bmod q$).

- For suitable primes $p$ we can compute $\Phi_\ell \bmod p$ in time $O(\ell^2 \log^3 p \operatorname{llog} p)$ using isogeny volcanoes [BLS 2011].
- Assuming the GRH, we can efficiently find sufficiently many such primes with $\log p = O(\log \ell)$.
- "Sufficiently many" is $O(\ell)$.

Henceforth we assume the GRH.

## Computing $\Phi_\ell$ with the CRT

Strategy: compute $\Phi_\ell \bmod p$ for sufficiently many primes $p$ and use the CRT to compute $\Phi_\ell$ (or $\Phi_\ell \bmod q$).

- For suitable primes $p$ we can compute $\Phi_\ell \bmod p$ in time $O(\ell^2 \log^3 p \operatorname{llog} p)$ using isogeny volcanoes [BLS 2011].
- Assuming the GRH, we can efficiently find sufficiently many such primes with $\log p = O(\log \ell)$.
- ”Sufficiently many” is $O(\ell)$.

Uses $O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$ expected time and $O(\ell^3 \log \ell)$ space.

Using the explicit CRT, we can directly compute $\Phi_\ell \bmod q$ using $O(\ell^2(n + \log \ell))$ space, where $n = \log q$.

But the size of $\phi_\ell$ is $O(\ell n)$.

Henceforth we assume the GRH.

## Computing $\phi_\ell(Y)$ with the CRT (naïve approach)

**Strategy:** lift $j(E)$ from $\mathbb{F}_q$ to $\mathbb{Z}$, compute $\Phi_\ell(X,Y) \bmod p$ and evaluate

$$\phi_\ell(Y) = \Phi_\ell(j(E), Y) \bmod p$$

for sufficiently many primes $p$. Obtain $\phi_\ell \bmod q$ via the explicit CRT.

Uses $O(\ell^2 \log^{3+\epsilon} p)$ expected time for each $p$, and $O(\ell^2 \log p)$ space.

## Computing $\phi_\ell(Y)$ with the CRT (naïve approach)

**Strategy:** lift $j(E)$ from $\mathbb{F}_q$ to $\mathbb{Z}$, compute $\Phi_\ell(X,Y) \bmod p$ and evaluate

$$\phi_\ell(Y) = \Phi_\ell(j(E),Y) \bmod p$$

for sufficiently many primes $p$. Obtain $\phi_\ell \bmod q$ via the explicit CRT.

Uses $O(\ell^2 \log^{3+\epsilon} p)$ expected time for each $p$, and $O(\ell^2 \log p)$ space.

However, “sufficiently many” is now $O(\ell n)$, where $n = \log q$.
Total expected time is $O(\ell^3 n \log^{3+\epsilon} \ell)$, using $O(\ell n + \ell^2 \log \ell)$ space.

This approach is **not very useful**:

- If $n$ is large (e.g. $n \approx \ell$), it takes way too long (quartic in $\ell$).
- It $n$ is small (e.g. $n \approx \log \ell$), it doesn’t save any space.

## Computing $\phi_\ell(Y)$ with the CRT (Algorithm 1)

Strategy: lift $j, j^2, j^3, \ldots, j^{\ell+1}$ from $\mathbb{F}_q$ to $\mathbb{Z}$ and then compute

$$\phi_\ell(Y) = \sum c_{ik} j^i Y^k \bmod p$$

for sufficiently many primes $p$, where $\Phi_\ell = \sum c_{ik} X^i Y^k$.
Obtain $\phi_\ell \bmod q$ via the explicit CRT.

## Computing $\phi_\ell(Y)$ with the CRT (Algorithm 1)

Strategy: lift $j, j^2, j^3, \ldots, j^{\ell+1}$ from $\mathbb{F}_q$ to $\mathbb{Z}$ and then compute

$$\phi_\ell(Y) = \sum c_{ik} j^i Y^k \bmod p$$

for sufficiently many primes $p$, where $\Phi_\ell = \sum c_{ik} X^i Y^k$.
Obtain $\phi_\ell \bmod q$ via the explicit CRT.

Now “sufficiently many” is $O(\ell + n)$.

For $n = O(\ell \log \ell)$, uses $O(\ell^3 \log^{3+\epsilon} \ell)$ expected time and $O(\ell^2 \log \ell)$ space (under GRH).
For $n = \Omega(\ell \log \ell)$, the space bound is optimal.

This algorithm can also evaluate the partial derivatives of $\Phi_\ell$ needed to construct normalized equations for $\tilde{E}$ (important for SEA).

## Computing $\phi_\ell(Y)$ with the CRT (Algorithm 2)

**Strategy:** lift $j(E)$ from $\mathbb{F}_q$ to $\mathbb{Z}$ and for sufficiently many primes $p$ compute $\phi_\ell \bmod p$ as follows:

1. For each of $\ell + 2$ $j$-invariants $y_i$, compute $z_i = \prod_k (j(E) - j_k)$, where the $j_k$ range over $\ell + 1$ neighbors of $y_i$ in $G_\ell(\mathbb{F}_p)$.
2. Interpolate $\phi_\ell(Y) \in \mathbb{F}_p$ as the unique polynomial of degree $\ell + 1$ for which $\phi_\ell(y_i) = z_i$.

Obtain $\phi_\ell \bmod q$ via the explicit CRT.

## Computing $\phi_\ell(Y)$ with the CRT (Algorithm 2)

**Strategy:** lift $j(E)$ from $\mathbb{F}_q$ to $\mathbb{Z}$ and for sufficiently many primes $p$ compute $\phi_\ell \bmod p$ as follows:

1. For each of $\ell+2$ $j$-invariants $y_i$, compute $z_i = \prod_k (j(E) - j_k)$, where the $j_k$ range over $\ell+1$ neighbors of $y_i$ in $G_\ell(\mathbb{F}_p)$.
2. Interpolate $\phi_\ell(Y) \in \mathbb{F}_p$ as the unique polynomial of degree $\ell+1$ for which $\phi_\ell(y_i) = z_i$.

Obtain $\phi_\ell \bmod q$ via the explicit CRT.

For $n = O(\ell^c)$, uses $O(\ell^3(n + \log\ell)\log^{1+\epsilon}\ell)$ expected time and $O(\ell n + \ell\log\ell)$ space (under GRH).

For $n = O(\log^{2-\epsilon} q)$ the algorithm is faster than computing $\Phi_\ell$.
For $n = \Omega(\log\ell)$ the space bound is optimal.

## Genus 1 point counting in large characteristic

Algorithms to compute $\#E(\mathbb{F}_q) = q + 1 - t$.

| Algorithm | Time | Space |
|---|---|---|
| Totally naive | $O(e^{2n+\epsilon})$ | $O(n)$ |
| Slightly less naive | $O(e^{n+\epsilon})$ | $O(n)$ |
| Baby-step giant-step | $O(e^{n/4+\epsilon})$ | $O(e^{n/4+\epsilon})$ |
| Pollard kangaroo | $O(e^{n/4+\epsilon})$ | $O(n^2)$ |
| Schoof | $O(n^5 \operatorname{llog} n)$ | $O(n^3)$ |
| SEA* | $O(n^4 \log^3 n \operatorname{llog} n)$ | $O(n^3 \log n)$ |
| SEA ($\Phi_\ell$ precomputed) | $O(n^4 \operatorname{llog} n)$ | $O(n^4)$ |

*Complexity estimates for SEA-based algorithms are heuristic expected times.

## Genus 1 point counting in large characteristic

Algorithms to compute $\#E(\mathbb{F}_q) = q + 1 - t$.

| Algorithm | Time | Space |
|---|---|---|
| Totally naive | $O(e^{2n+\epsilon})$ | $O(n)$ |
| Slightly less naive | $O(e^{n+\epsilon})$ | $O(n)$ |
| Baby-step giant-step | $O(e^{n/4+\epsilon})$ | $O(e^{n/4+\epsilon})$ |
| Pollard kangaroo | $O(e^{n/4+\epsilon})$ | $O(n^2)$ |
| Schoof | $O(n^5 \operatorname{llog} n)$ | $O(n^3)$ |
| SEA* | $O(n^4 \log^3 n \operatorname{llog} n)$ | $O(n^3 \log n)$ |
| SEA ($\Phi_\ell$ precomputed) | $O(n^4 \operatorname{llog} n)$ | $O(n^4)$ |
| | | |
| SEA with Algorithm 1 | $O(n^4 \log^2 n \operatorname{llog} n)$ | $O(n^2 \log n)$ |
| Amortized | $O(n^4 \operatorname{llog} n)$ | $O(n^2 \log n)$ |

*Complexity estimates for SEA-based algorithms are heuristic expected times.

## Alternative modular polynomials

In practice, the modular polynomials $\Phi_\ell$ are not used in SEA. There are alternatives (due to Atkin, Müller, and others) that are smaller by a large constant factor (100x to 1000x is typical).

The isogeny-volcano approach of [BLS 2010] can compute many types of (symmetric) modular polynomials derived from modular functions other than $j(z)$, but these do not include the modular polynomials commonly used with SEA.

## Alternative modular polynomials

In practice, the modular polynomials $\Phi_\ell$ are not used in SEA. There are alternatives (due to Atkin, Müller, and others) that are smaller by a large constant factor (100x to 1000x is typical).

The isogeny-volcano approach of [BLS 2010] can compute many types of (symmetric) modular polynomials derived from modular functions other than $j(z)$, but these do not include the modular polynomials commonly used with SEA.

They do include modular polynomials $\Phi_\ell^{\mathfrak{f}}$ derived from the Weber function $\mathfrak{f}(z)$, but these have never (?) been used with SEA before.

Provided $\mathrm{End}(E)$ has discriminant $D \equiv 1 \bmod 8$ with $3 \nmid D$, the polynomial $\phi_\ell^{\mathfrak{f}}(Y) = \Phi_\ell^{\mathfrak{f}}(\mathfrak{f}(E), Y)$ parameterizes $\ell$-isogenies from $E$.

This condition is easily checked (without knowing $D$).
If it fails, powers of $\mathfrak{f}$, or other modular functions may be used.

## The Weber function

The Weber $\mathfrak{f}$-function is defined by

$$\mathfrak{f}(\tau) = \frac{\eta\big((\tau+1)/2\big)}{\zeta_{48}\eta(\tau)},$$

and satisfies $j(\tau) = (\mathfrak{f}(\tau)^{24} - 16)^3/\mathfrak{f}(\tau)^{24}$.

The coefficients of $\Phi_\ell^{\mathfrak{f}}$ are roughly 72 times smaller.
This means we need 72 times fewer primes.

The polynomial $\Phi_\ell^{\mathfrak{f}}$ is roughly 24 times sparser.
This means we need 24 times fewer interpolation points.

Overall, we get nearly a **1728-fold speedup** using $\Phi_\ell^{\mathfrak{f}}$.

## Modular polynomials for $\ell = 11$

Classical:

$$X^{12} + Y^{12} - X^{11}Y^{11} + 8184X^{11}Y^{10} - 28278756X^{11}Y^{9} + 53686822816X^{11}Y^{8}$$
$$- 61058988656490X^{11}Y^{7} + 42570393135641712X^{11}Y^{6} - 17899526272883039048X^{11}Y^{5}$$
$$+ 4297837238774928467520X^{11}Y^{4} - 529134841844639613861795X^{11}Y^{3} + 27209811658056645815522600X^{11}Y^{2}$$
$$- 374642006356701393515817612X^{11}Y + 296470902355240575283200000X^{11}$$

. . . 8 pages omitted . . .

$$+ 392423345094527654908696462408720049099524723370674627089936420642670174061941686739245465600 0 \ldots 000$$

Atkin:

$$X^{12} - X^{11}Y + 744X^{11} + 196680X^{10} + 187X^{9}Y + 21354080X^{9} + 506X^{8}Y + 830467440X^{8}$$
$$- 11440X^{7}Y + 16875327744X^{7} - 57442X^{6}Y + 208564958976X^{6} + 184184X^{5}Y + 1678582287360X^{5}$$
$$+ 1675784X^{4}Y + 9031525113600X^{4} + 1867712X^{3}Y + 32349979904000X^{3} - 8252640X^{2}Y + 74246810880000X^{2}$$
$$- 19849600XY + 98997734400000X + Y^{2} - 8720000Y + 58411072000000$$

Weber:

$$X^{12} + Y^{12} - X^{11}Y^{11} + 11X^{9}Y^{9} - 44X^{7}Y^{7} + 88X^{5}Y^{5} - 88X^{3}Y^{3} + 32XY$$

## Elliptic curve point counting record

The number of points on the elliptic curve $E$ defined by

$$y^2 = x^3 + 2718281828x + 3141592653,$$

modulo the 5011 digit prime $q = 16219299585 \cdot 2^{16612} - 1$ is

8323769891444946600619018491391378260069836370604500159309667928183741136740938227669912830997846627009617004020582940190774831705166648378125548174433501
6222360544000538839492022451911485986733819166009550859216525385267852842524240978796544500427958734245859103650693623260065854955676905842760404211102908
0666232135885662070661039670759580341918109430064160840690748363019037103169978894180556726367014400296781983798513562269371401276427209286702254047174078
4700901798590441199208750379215971112344019653309999668029194772178482699210001668960742884085944350942098735441112464897682811881029409157742761498481361
8236133983076302692999418138548552140105778012525989072405641889553339872433242793570967700290860169473820597303300518069505065832587533308670748048008463
6983900427134645786532440716786520228221019906549532681092997885462429828848191629734823903084330670554604329550248173093287043318053279349574487888250634
8393787807087351238867988051327037590331790801872453585872437467694874112726738073095037665888862659824866162979710551480066332118269833639587932989704356
2635494364468486039656664278370935750099790919223024134537160958876614320893731637296530257682556027127545666105422232328156220481118882835904832158925287
2815308704965441879416303457576489111718650037380917938646571605607395885788665998491783840002043757298666639706781737384345665795929791423993337711367782
2538016636015241053779745447935639933068437226703067711612870475974728874060256153829424355309461429412863767016010448708725782340275978368434887328902748
7046203327761442798102604298830732855895993246333047147994546492842426742530314565704272126471147496267335652133743455000287920232413723922825839150351274
2950736034773585892234313092778077346572608561779267925193030390180619815277308025700377636113052880114730236382334852026580407537787327013748289451197304
6679428777685534275306203922096387437778405610945156936507440996084119730814303901482626498520813641540064044431078342859539882090926223504237272408488115
4327002269478397116252120617133360022725560655793168849910978673768497963315764527084692590231159741512227876106228667690675220660368352958211682369185130
5917272462661882973355576998865646958429361081809162692181866270338066704102681199813126843679500076625472860409064749186815445274367067586843407055463402
1891358398165724885432541336551115909636456700693477449865260368511045411205847035453060636486512589217930914523201112950463798415869418991750541041378713
1053621887908831837273005465881200616271744801682487747455898185251772280214510451501147795355549876845352299986818351761176510147685763441401085581041504
5320737093505215091386326150432421200754980473235846455348856098791946896114485582546561261445641145852160774738994958659512607430608581213461723630995676
5621860727156856482046150112012015113007122286669299590274282107690933890300810525603558710045399536727403969324912420806389527152955993943311641026894824
2637362685355343705851022198361778056561670358618600362869633742502575881826442008352428041311901722720144145496595474571658798526002326473049911956443052
8793982788520765498812575122123974107524497342781984377646508955766617133740349759685046139983532743454125015169809484605689099695691493817145995186900066
9270969498525993914751206760782244790625122626848753012733495289200641795967184561011226439252930596090031649974277634943933178938510726596259437826466293
3791621325648258956921290293302567147491547700031400327806411025863588895745234111758218534120042661084581341547473844325846515861998968494758420093653389
5253588411116027196086899901474201259197147823729252481394860002922779612554902938159895772157480500747896699702410869127401835778517148930637715216609619
1664750803979956621679571978953552211724552632230710653244433669331067442040140391602456581858747401436772403284080454895800825555079522458369190254711040
6012000284990126494269674951154806364097330589798793851739761556415874133478898662870219506352034178937096525462482561334178354529257317157406885610633216
4105705546182508456321207036745733148635468184175804925273259911659543081743640608001131591890082864513124247013731366137271496104705809743302158675109390
8895745441684195336715770412686321350796787391486224738612016911715739107481092463845433183146882764205897555692474146749724490484592370638492569933497005
0287810480327348970976228332956433891007862170869157707253005290879107235507514013187875766473645717669386184406205549908141812073945776391634588349492676
8779319474713905005440022152143445859931404486738106328557238092332971352015340561574971125269604744349474765249731664806205766695327115242584334178977061
6064557079435236406353029020560141015314103419765029534921177065625577774688408769857858804251711896591035794406728856602939161842221528770720582112364327
7156318567658978483022412314216285445946753013230236619421604149931783961968774559963411288277953692794747738279937358682979368994295124969120288710932706
32846246774367220129816851945807778140092913366453585259624246494437340122223955248

## Elliptic curve point counting record

| Task | Total CPU Time |
|---|---:|
| Compute $\phi_\ell^{\mathfrak{f}}$ | 32 days |
| Find a root $\tilde{j}$ | 995 days |
| Compute $g_\ell$ | 3 days |
| Compute $\pi \bmod g_\ell, E$ | 326 days |
| Find $\lambda_\ell$ | 22 days |

$\phi_\ell^{\mathfrak{f}}(Y) = \Phi_\ell^{\mathfrak{f}}(j(E), Y)$ was computed for $\ell$ from 5 to 11681.
Exactly 700 of 1400 were found to be Elkies primes.
Atkin primes were not used.

The largest $\phi_\ell^{\mathfrak{f}}$ was under 20MB in size and took about two hours to compute using 1 core.

## Modular polynomial evaluation record

For $\ell = 100019$ and $q = 2^{86243} - 1$ we computed $\phi_\ell^{\mathfrak{f}}(Y) = \Phi_\ell^{\mathfrak{f}}(j(E), Y)$.

This is much larger than one would need to set a 25,000 digit point-counting record.

The size of $\phi_\ell^{\mathfrak{f}}$ is about **1 GB**.

## Modular polynomial evaluation record

For $\ell = 100019$ and $q = 2^{86243} - 1$ we computed $\phi_\ell^{\mathfrak{f}}(Y) = \Phi_\ell^{\mathfrak{f}}(j(E), Y)$.

This is much larger than one would need to set a 25,000 digit point-counting record.

The size of $\phi_\ell^{\mathfrak{f}}$ is about **1 GB**.

For comparison:

- The size of $\Phi_\ell^{\mathfrak{f}} \bmod q$ is about **2 TB**.

## Modular polynomial evaluation record

For $\ell = 100019$ and $q = 2^{86243} - 1$ we computed $\phi_\ell^{\mathfrak{f}}(Y) = \Phi_\ell^{\mathfrak{f}}(j(E), Y)$.

This is much larger than one would need to set a 25,000 digit point-counting record.

The size of $\phi_\ell^{\mathfrak{f}}$ is about **1 GB**.

For comparison:

- The size of $\Phi_\ell^{\mathfrak{f}} \bmod q$ is about **2 TB**.
- The size of $\Phi_\ell \bmod q$ is about **50 TB**.

## Modular polynomial evaluation record

For $\ell = 100019$ and $q = 2^{86243} - 1$ we computed $\phi_\ell^{\mathfrak{f}}(Y) = \Phi_\ell^{\mathfrak{f}}(j(E), Y)$.

This is much larger than one would need to set a 25,000 digit point-counting record.

The size of $\phi_\ell^{\mathfrak{f}}$ is about **1 GB**.

For comparison:

- The size of $\Phi_\ell^{\mathfrak{f}} \bmod q$ is about **2 TB**.
- The size of $\Phi_\ell \bmod q$ is about **50 TB**.
- The size of $\Phi_\ell$ is more than **10 PB**.

## Improved space complexity of computing horizontal isogenies

The algorithm of [Bisson-S 2011] for computing the endomorphism ring of an elliptic curve $E/\mathbb{F}_q$ runs in $L[1/2, \sqrt{3}/2]$ expected time and uses $L[1/2, 1/\sqrt{3}]$ space (under GRH).

The space complexity can now be improved to $L[1/2, 1/\sqrt{12}]$.

A similar improvement applies to algorithms for computing horizontal isogenies of large degree [Jao-Soukharev ANTS IX].