Fast computation of modular polynomials

Andreas Enge

Isogeny party

Palaiseau, July 18, 2006

enge@lix.polytechnique.fr

http://www.lix.polytechnique.fr/Labo/Andreas.Enge

INRIA Futurs & Laboratoire d'Informatique (LIX)
École polytechnique
France

Modular polynomial computation

Modular functions

- Γ' a subgroup of finite index of $\Gamma = \mathrm{Sl}_2(\mathbb{Z})$
- $f : \mathbb{H} = \{z \in \mathbb{C} : \Im z > 0\} \to \mathbb{C}$ is modular for Γ' if it is
  - meromorphic
  - invariant under Γ':

$$f(Mz) = f\left(\frac{az+b}{cz+d}\right) = f(z) \text{ for } M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma'$$

  - meromorphic at infinity; for some N,

$$q = e^{2\pi i z}, f(z) = \sum_{\nu=\nu_0}^{\infty} a_\nu q^{\nu/N}$$

  - meromorphic at the other cusps ...

Modular functions for Γ

- $\Gamma = \langle T, S \rangle$
  - $T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $Tz = \frac{1 \cdot z + 1}{0 \cdot z + 1} = z + 1$
  - $S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, $Sz = \frac{0 \cdot z - 1}{1 \cdot z + 0} = \frac{-1}{z}$
- $\mathbb{C}_\Gamma = \mathbb{C}(j)$
- $j(z) = q^{-1} + 744 + \cdots$

Modular functions for $\Gamma^0(\ell)$

- $\Gamma^0(\ell) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma : \ell | b \right\}$
- $T^\ell = \begin{pmatrix} 1 & \ell \\ 0 & 1 \end{pmatrix} \in \Gamma^0(\ell), \quad T^\ell z = z + \ell$
- $\mathbb{C}_{\Gamma^0(\ell)} = \mathbb{C}\left(j(z), j(z/\ell)\right)$
- $\mathbb{C}_{\Gamma^0(\ell)} = \mathbb{C}\left(j(z), \mathfrak{w}_\ell^{2s}(z)\right)$ with $\mathfrak{w}_\ell = \frac{\eta(z/\ell)}{\eta(z)}$, $s = 12/\gcd(12, \ell - 1)$

Modular polynomials

- $$\Phi(X) = \prod_{\nu=1}^{[\Gamma:\Gamma']} (X - f(M_\nu z)) = X^{[\Gamma:\Gamma']} + \sum_{r=1}^{[\Gamma:\Gamma']} c_r X^{[\Gamma:\Gamma']-r}$$
  - M_ν representatives of $\Gamma' \backslash \Gamma$
  - $c_r \in \mathbb{Z}[j]$
- THE modular polynomial for $\Gamma^0(\ell)$

$$\begin{aligned} \Phi_\ell(X) &= \prod_{\nu=0}^{\ell-1} (X - j(T^\nu z/\ell))(X - j(Sz/\ell)) \\ &= \prod_{\nu=0}^{\ell-1} (X - j((z+\nu)/\ell))(X - j((-1/z)/\ell)) \end{aligned}$$

Example: $\ell = 5$

$$\begin{aligned} \Phi_5 \;=\; & X^6 - X^5 j^5 + 3720 X^5 j^4 - 4550940 X^5 j^3 + 2028551200 X^5 j^2 \\ & - 246683410950 X^5 j + 1963211489280 X^5 + 3720 X^4 j^5 \\ & + 1665999364600 X^4 j^4 + 10787892818533680 0 X^4 j^3 \\ & + 38308360977981121537 5 X^4 j^2 + 12854179890682881638400 0 X^4 j \\ & + 12847331328414244562534 40 X^4 - 4550940 X^3 j^5 \\ & + 10787892818533680 0 X^3 j^4 - 44120696551291483524610 0 X^3 j^3 \\ & + 26898488858380731577417728000 X^3 j^2 \\ & - 19245793461892829965510823116800 0 X^3 j \\ & + 28024477782843952780432156529786880 0 X^3 + 2028551200 X^2 j^5 \\ & + 38308360977981121537 5 X^2 j^4 \\ & + 26898488858380731577417728000 X^2 j^3 \\ & + \cdots + 14135994715472135869775347469107136275100467200 0 \end{aligned}$$

General modular polynomials

Between any two modular functions f and g there is a modular polynomial $\Phi \in \mathbb{C}[X, Y]$ s.t. $\Phi(f, g) = 0$.

Height

- Theorem (P. Cohen 1984):

$$h(\Phi_\ell) = 6(\ell+1)\big(\log \ell - 2\log \ell/\ell + O(1)\big)$$

- Φ_ℓ has bit size $\tilde{\Theta}(\ell^3)$

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

Applications

Φ_ℓ parameterises cyclic isogenies of degree ℓ of elliptic curves:

$$\Phi_\ell(j,\tilde{j}) = 0 \Leftrightarrow E \text{ and } \tilde{E} \text{ are } \ell\text{-isogenous}$$

- SEA
- cryptography — hash functions, ElGamal
- class invariants for complex multiplication:

$$\frac{\eta(z/p_1)\eta(z/p_2)}{\eta(z/p_1p_2)\eta(z)} \text{ and } j(z)$$

- p-adic algorithm for class polynomials:

$$\frac{\eta(z/p_1)\eta(z/p_2)}{\eta(z/p_1p_2)\eta(z)} \text{ and the same in } z/\ell$$

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

Idea of the algorithm

$$\Phi_\ell(X) = \prod_{\nu=0}^{\ell-1}(X - j((z+\nu)/\ell))(X - j((-1/z)/\ell))$$

- $j(z) = q^{-1} + 744 + 196884q + 21493760q^2 + 864299970q^3 + \cdots$
- $q((z+\nu)/\ell) = \zeta_\ell^\nu q^{1/\ell}$
- $q((-1/z)/\ell) =$???
- $j(-1/(z\ell)) = j(\ell z)$
- $\mathfrak{w}_\ell(-1/z) = \sqrt{\ell}\,\frac{\eta(\ell z)}{\eta(z)}$

Handling the non-controversial part

$$\tilde{\Phi}_\ell(X) = \prod_{\nu=0}^{\ell-1}(X - j((z+\nu)/\ell))$$

- $s_r = \sum_{\nu=0}^{\ell-1} j((z+\nu)/\ell)^r$
- $j(z)^r = \sum_{k=-r}^{\infty} a_{k,r} q^k$
- $j((z+\nu)/\ell)^r = \sum_{k=-r}^{\infty} a_{k,r}\zeta_\ell^{k\nu} q^{k/\ell}$
- $s_r = \sum_{k=0}^{\infty} a_{k\ell,r} q^k$
- Newton's formulæ

Complexity

- s_r up to q^d
- j^r up to $q^{\ell d}$
- ℓ multiplications of series with $O(\ell d)$ terms at precision $O(\ell \log \ell)$

$$\tilde{O}(\ell^3 d)$$

- $j(\ell z) = q^{-\ell} + \cdots \Rightarrow d = \ell$

$$\tilde{O}(\ell^4)$$

- Remark: modulo p in

$$\tilde{O}(\ell^3 \log p)$$

Shortcomings

- special role of S
- composite level ...

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

Charles–Lauter

Aim: $\Phi_\ell \bmod p$

- uses moduli interpretation of Φ_ℓ: ℓ-isogenies!
- $\overline{j} \in \mathbb{F}_{p^2}$ supersingular j-invariant
- determine ℓ^2 torsion points in extension of degree $O(\ell)$
- compute isogenous curves via Vélu's formulæ
- yields $\Phi_\ell(X, \overline{j})$
- compute $\ell+1$ instantiations and interpolate

$$\tilde{O}(\ell^4 \log^2 p) \subseteq \tilde{O}(\ell^6)$$

- compare with q-expansions

$$\tilde{O}(\ell^3 \log p) \subseteq \tilde{O}(\ell^4)$$

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

Andreas Enge

19

Algorithm (Borwein–Borwein 1987)

$$\Phi(X) = \prod_{\nu=1}^{[\Gamma:\Gamma']} (X - f(M_\nu z)) = X^{[\Gamma:\Gamma']} + \sum_{r=1}^{[\Gamma:\Gamma']} c_r(j(z)) X^{[\Gamma:\Gamma']-r}$$

- Evaluate

$$\Phi(X)(z_k) = \prod_{\nu=1}^{[\Gamma:\Gamma']} (X - f(M_\nu z_k)) = X^{[\Gamma:\Gamma']} + \sum_{r=1}^{[\Gamma:\Gamma']} c_r(j(z_k)) X^{[\Gamma:\Gamma']-r}$$

  for $\deg_j \Phi + 1$ values $z_k \in \mathbb{H}$

- Interpolate

$$c_r = \sum_{\nu=0}^{\deg_j \Phi} c_{r,\nu} j^{\deg_j \Phi - \nu} \in \mathbb{Z}[j]$$

  whose values are known in the $j(z_k)$

Complexity

- $O(\ell)$ evaluations of j for $O(\ell)$ values z_k at precision $O(\ell \log \ell)$

$$\boxed{O(\ell^2 E(\ell \log \ell))}$$

- $O(\ell)$ computations of polynomials of degree $O(\ell)$ from its roots

$$\boxed{O(\ell \cdot \ell \log \ell M(\ell \log \ell) \cdot \log \ell)}$$

- $O(\ell)$ interpolations of polynomials of degree $O(\ell)$

same

$$\boxed{O\big(\ell^2\big(\log^2 M(\ell \log \ell) + E(\ell \log \ell)\big)\big)}$$

$$E(n) = O(\log n M(n))$$

$$\boxed{O(\ell^3 \log^4 \ell \log \log \ell) = \tilde{O}(\ell^3)}$$

Advantages

- complexity
- easy to implement
- works out of the box for arbitrary Γ'
  no hassle with q-expansions at the cusps

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

Reference

A. Enge 2006:

"The complexity of class polynomial computation via floating point approximations"

`http://hal.inria.fr/inria-00001040`

`http://arxiv.org/abs/cs.CC/0601104`

Sparse series for η

- $\eta = q^{1/24}\left(1+\sum_{k=1}^{\infty}(-1)^k\left(q^{k(3k-1)/2}+q^{k(3k+1)/2}\right)\right)$
- $f_1 = \dfrac{\eta\left(\frac{z}{2}\right)}{\eta(z)}$
- $j = \dfrac{(f_1^{24}+16)^3}{f_1^{24}}$
- sparse series — $O(\sqrt{n})$ terms yield precision of $O(n)$ digits
- exponents are values of polynomials — iterated differences
- one new term with $O(1)$ multiplications

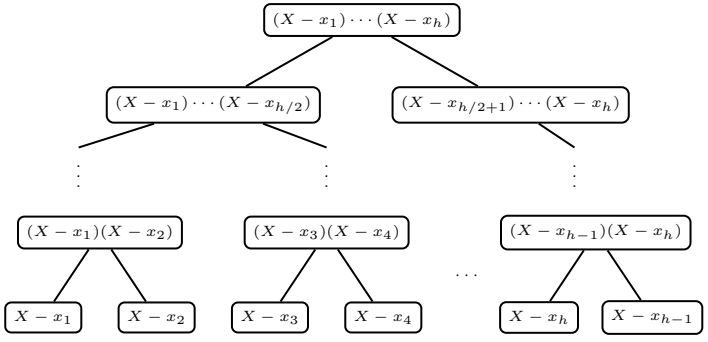
$$E(n) = O\left(\sqrt{n}M(n)\right)$$

- q-expansions are beaten

$$\tilde{O}\left(\ell^{3.5}\right)$$

Multievaluation

- Task: evaluate a polynamial η or j of degree $O(n)$ in $O(n)$ points x_k
- Fundamental observation: $\eta(x_k) = \eta \bmod (X - x_k)$



$$E(n) = \frac{n\log^2 nM(n)}{n}$$

AGM

Th. (Régis Dupont 2006):
One can evaluate any modular function at precision n in time

$$E(n) = O(\log n\, M(n)) = \tilde{O}(n).$$

Idea
Newton iterations
on a function involving the
arithmetic-geometric mean (AGM)

Theta constants — definition

$$\begin{aligned}
\theta_{00}(z) &= \sum_{n\in\mathbb{Z}} q^{n^2/2} = 1 + 2q^{1/2} + 2q^2 + 2q^{9/2} + \dots \\
\theta_{01}(z) &= \sum_{n\in\mathbb{Z}} (-1)^n q^{n^2/2} = 1 - 2q^{1/2} + 2q^2 - 2q^{9/2} + \dots \\
\theta_{10}(z) &= \sum_{n\in\mathbb{Z}} q^{(n+1/2)^2/2} = q^{1/8} + 2q^{9/8} + 2q^{25/8} + \dots
\end{aligned}$$

Arithmetic–geometric mean

$$\begin{aligned}\theta_{00}^2(2z) &= \frac{\theta_{00}^2(z)+\theta_{01}^2(z)}{2}\\ \theta_{01}^2(2z) &= \sqrt{\theta_{00}^2(z)\theta_{01}^2(z)}\end{aligned}$$

- AGM for $a, b \in \mathbb{C}$
  - $a_0 = a$, $b_0 = b$
  - $a_{n+1} = \frac{a_n+b_n}{2}$
  - $b_{n+1} = \sqrt{a_n b_n}$
  - converges quadratically to a common limit $\mathrm{AGM}(a, b)$
- $\mathrm{AGM}(a, b) = a \cdot \mathrm{AGM}(1, b/a)$; $M(x) := \mathrm{AGM}(1, x)$

The modular functions k and k'

- $k(z) = \left(\frac{\theta_{10}(z)}{\theta_{00}(z)}\right)^2$
- $k'(z) = \left(\frac{\theta_{01}(z)}{\theta_{00}(z)}\right)^2$
- $k^2(z) + k'^2(z) = 1$
- $j = 256\frac{(1-k'^2+k'^4)^3}{k'^4(1-k'^2)^2}$
- $\eta^{12} = \frac{k'^2(1-k'^2)}{16M^6(k')}$
- $M(k'(z)) = \frac{1}{\theta_{00}^2(z)}$

Newton iterations

- $f_z(x) = iM(x) - zM(\sqrt{1-x^2})$
- $f_z(k'(z)) = 0$
- $g_z(x) = \frac{-2M(x)^3}{\pi z x(1-x^2)}$
- $g_z(k'(z)) = f_z'(k'(z))$

$$x_{n+1} \leftarrow x_n - \frac{f_z(x_n)}{g_z(x_n)}$$

converges quadratically to $k'(z)$

Complexity

- Newton
  - increasing precision, only the last iteration counts
- AGM
  - quadratic convergence $\Rightarrow O(\log n)$ steps for $O(n)$ digits
  - each step in $O(M(n))$
  - total $O(\log n M(n))$

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

Implementation

- in C using MPC, MPFR, GMP
- tricks
  - $z_k \in i\mathbb{R} \Rightarrow q(z_k) \in \mathbb{R} \Rightarrow j(z_k) \in \mathbb{R}$
  - $j(z_k) = 1728 + k$, simplifies interpolation (R. Dupont)
  - parallelisation controlled via files and shell scripts
  - matrix transposition on disk
- non-tricks
  - η-evaluation via slow series
  - slow interpolation by iterated differences

Example for SEA: $\ell = 211$

$$f = \frac{T_r\big(\eta(\ell z)\eta(z)\big)}{\eta(\ell z)\eta(z)}$$

- $r = 13$, $\deg_f = 212$, $\deg_j = 16$
- precision 696 bits
- time for evaluation (Pentium-M 1.8 GHz):
  $17 \cdot 1.4$ s $= 24$ s
- time for interpolation:
  0.1 s

Example for SEA: $\ell = 10079$

- $r = 5$, $\deg_f = 10080$, $\deg_j = 672$
- precision 35051 bits
- time for evaluation (Athlon-64 2.4 GHz):
  $673 \cdot 15400$ s ≈ 120 d
- time for interpolation:
  56300 s ≈ 16 h
- size: ≈ 16 GB

A. E., P. Gaudry, F. Morain 2005:

`The cardinality of the curve` $E : Y^2 = X^3 + 4589X + 91128$
`modulo` $p = 10^{2004} + 4863$ `is p+1-t with`
`t=-145747366218116724980830773581217171221551221862716051`
`...52835836543869391164614140827319838307446969929268`

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

Example: $\ell = 5$

$$\begin{aligned}\Phi_5 &= X^6 - X^5j^5 + 3720X^5j^4 - 4550940X^5j^3 + 2028551200X^5j^2\\ &\quad -246683410950X^5j + 1963211489280X^5 + 3720X^4j^5\\ &\quad +1665999364600X^4j^4 + 10787892818533680 0X^4j^3\\ &\quad +38308360977981121537 5X^4j^2 + 128541798906828816384000X^4j\\ &\quad +12847331328414244562534 40X^4 - 4550940X^3j^5\\ &\quad +10787892818533680 0X^3j^4 - 44120696551291483524610 0X^3j^3\\ &\quad +26898488858380731577417728000X^3j^2\\ &\quad -19245793461892829965510823116 8000X^3j\\ &\quad +28024477782843952780432156529786 8800X^3 + 2028551200X^2j^5\\ &\quad +38308360977981121537 5X^2j^4\\ &\quad +\cdots+ 14135994715472135869775347469107136275100467 2000\\ \Phi_5^{\mathsf{Sch}} &= X^6 - f^5X^5 + 4fX + f^6\end{aligned}$$

Definition

$$f = \zeta_{48}^{-1}\frac{\eta\left(\frac{z+1}{2}\right)}{\eta(z)}$$

$$\Phi_\ell^{\mathsf{Sch}}(f(z), f(z/\ell)) = 0$$

Algorithm

- $f(z)$ is modular for $\Gamma(48)$, $g(z) = f(z/\ell)$ for $\Gamma^0(\ell)\cap\Gamma(48)$

$$\Phi_\ell^{\mathsf{Sch}}(X) = \prod_{\nu=0}^{\ell}(X - g(M_\nu z)) = X^{\ell+1} + \sum_{r=1}^{\ell+1} c_r X^{\ell+1-r}$$

- $c_r \in \mathbb{Z}[f]$
- M_ν representatives of $\Gamma^0(\ell)\cap\Gamma(48)\backslash\Gamma(48)$:

$$\begin{pmatrix}1 & 48\nu\\ 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}1-48k & 48k\\ -48k & 1+48k\end{pmatrix}, \quad k = 48^{-1} \bmod \ell$$

Example: $\ell = 211$

- $\deg_f = \deg g = 212$
- precision 2277 bits
- time for evaluation (Pentium-M 1.8 GHz):
  $213 \cdot 1.5$ s ≈ 5 min
- time for interpolation:
  12 s

Modular polynomial computation

1. Modular polynomials
2. Applications
3. q-expansions
4. Charles–Lauter
5. Evaluation–interpolation
6. Fast evaluation of modular functions
7. Implementation
8. Schläfli equations
9. Generalised Schläfli equations

$$f = \mathfrak{w}_p = \frac{\eta\left(\frac{z}{p}\right)}{\eta(z)}$$

$$f = \mathfrak{w}_{p_1,p_2} = \frac{\eta\left(\frac{z}{p_1}\right)\eta\left(\frac{z}{p_2}\right)}{\eta\left(\frac{z}{p_1p_2}\right)\eta(z)}$$

- Wanted: modular polynomial between $f(z)$ and $f(z/\ell)$
- Algorithm works . . .
- . . . if polynomial exists (Hauptmodul)!