Class polynomials for abelian surfaces

Andreas Enge

LFANT project-team
INRIA Bordeaux–Sud-Ouest
andreas.enge@inria.fr
http://www.math.u-bordeaux.fr/~aenge

LFANT seminar
27 January 2015
(joint work with Emmanuel Thomé)
Elliptic curves
- Applications of elliptic curves
- Complex multiplication theory
- Algorithms

Abelian surfaces
- Complex multiplication theory
- Algorithms
- Implementation and examples
Elliptic curves

- \( E : Y^2 = X^3 + aX + b, \quad a, b \in \mathbb{F}_p \)
- Abelian variety of dimension 1 \( \Rightarrow \) finite group

- Hasse 1934
  \[ |\#E(\mathbb{F}_p) - (p + 1)| \leq 2\sqrt{p} \]
- Moduli space of dimension 1 parameterised by invariant

\[
j = \frac{1728}{4a^3 + 27b^2}
\]
If $P \in E(\mathbb{Z}/N_1 \mathbb{Z})$ with $P$ of prime order $N_2$, 

$$N_2 > \left(\frac{4\sqrt{N_1} + 1}{\sqrt{N_1}}\right)^2,$$

then $N_1$ is prime.

Record: 25 050 decimal digits (Morain 2010)
Cryptography

- Discrete logarithm based cryptography
  - Need prime cardinality
  - Prefer random curves

- Pairing-based cryptography Weil and (reduced) Tate pairing

\[
e : E(\mathbb{F}_p)[\ell] \times E(\mathbb{F}_{p^k})[\ell] \rightarrow \mathbb{F}_{p^k}^\times[\ell]
\]

- Bilinear: \( e(aP, bQ) = e(P, Q)^{ab} \)
- An exponential number of cryptographic primitives...
- Need CM constructions for suitable curves.
1. **Elliptic curves**
   - Applications of elliptic curves
   - *Complex multiplication theory*
   - Algorithms

2. **Abelian surfaces**
   - *Complex multiplication theory*
   - Algorithms
   - Implementation and examples
Deuring 1941: The endomorphism ring of an (ordinary) elliptic curve is either \( \mathbb{Z} \), or an order

\[
\mathcal{O}_D = \left[ 1, \frac{D + \sqrt{D}}{2} \right] \mathbb{Z}
\]

of discriminant \( D < 0 \) in \( K = \mathbb{Q}(\sqrt{D}) \).

<table>
<thead>
<tr>
<th>( E ) with complex multiplication by ( \mathcal{O}_D ) / by ( D )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over ( \mathbb{C} ): usually ( \mathbb{Z} ), sometimes ( \mathcal{O}_D )</td>
</tr>
<tr>
<td>Over ( \mathbb{F}_p ): always ( \mathcal{O}_D )!</td>
</tr>
</tbody>
</table>
Complex multiplication

- **Frobenius:** \( \pi : (x, y) \mapsto (x^p, y^p) \), fixes \( E(\mathbb{F}_p) \)

- **Deuring 1941:** Any (ordinary) curve over \( \mathbb{F}_p \) is the reduction of a curve over \( \mathbb{C} \) with the same endomorphism ring.

- **Hasse:** \( \pi = \frac{t + \sqrt{D}}{2}, \quad \text{Tr}(\pi) = t, \quad N(\pi) = \frac{t^2 - D}{4} = p \)

  \[ \#E(\mathbb{F}_p) = p + 1 - t \]
Given $D$, what are the curves over $\mathbb{C}$ with CM by $D$?

- Modular invariant
  $$j : \mathbb{H} = \{ z \in \mathbb{C} : \Im(z) > 0 \} \rightarrow \mathbb{C}$$

- $\varphi : K = \mathbb{Q}(\sqrt{D}) \rightarrow \mathbb{C}$ embedding

- $\mathfrak{a} = (\alpha_1, \alpha_2)$ ideal of $\mathcal{O}_D$ with basis quotient $\tau = \varphi \left( \frac{\alpha_2}{\alpha_1} \right) \in \mathbb{H}$

- $j(\tau)$ depends only on the ideal class of $\mathfrak{a}$; determines the $h = \# \text{Cl}(\mathcal{O}_D)$ curves with CM by $D$. 
First main theorem of complex multiplication

\[ \Omega_D = K(j(a)) \]

\[ K = \mathbb{Q}(\sqrt{D}) \]

\[ \mathbb{Q} \]

\[ \Omega_D = \text{Hilbert class field of } K \text{ (for } D \text{ fundamental discriminant)} \]

= maximal abelian, unramified extension of \( K \)

\[ \sigma : \text{Cl}(O_D) \xrightarrow{\sim} \text{Gal}(\Omega_D/K) \]

\[ j(a)^{\sigma(b)} = j(ab^{-1}) \]
1 Elliptic curves
   - Applications of elliptic curves
   - Complex multiplication theory
   - Algorithms

2 Abelian surfaces
   - Complex multiplication theory
   - Algorithms
   - Implementation and examples
Main algorithm

- Fix $D < 0$ and $p$ prime s.t. $p = \frac{t^2 - \sqrt{D}}{4}$ and $N = p + 1 - t$ convenient
- Enumerate the $h$ ideal classes of $\mathcal{O}_D$:
  \[
  \left( A_i, \frac{-B_i + \sqrt{D}}{2} \right)
  \]
- Compute over $\mathbb{C}$ the class polynomial
  \[
  H(X) = \prod_{i=1}^{h} \left( X - j \left( \frac{-B_i + \sqrt{D}}{2A_i} \right) \right) \in \mathbb{Z}[X]
  \]
- Find a root $\bar{j}$ modulo $p$
- Write down the curve $E : Y^2 = X^3 + aX + b$ with
  \[
  c = \frac{\bar{j}}{1728 - \bar{j}}, \quad a = 3c, \quad b = 2c
  \]
Complexity

- **Size of** $H$
  - Degree $h \in O\left(\sqrt{|D|}\right)$ (Littlewood 1928)
  - Coefficients with $O\left(\sqrt{|D|}\right)$ digits (Schoof 1991, E. 2009)
  - Total size $O\left(|D|\right)$

- **Evaluation of** $j$: $O\left(\sqrt{|D|}\right)$
  - Precision: $O\left(\sqrt{|D|}\right)$ digits
  - Multievaluation of the “polynomial” $j$ (E. 2009)
  - Arithmetic-geometric mean (Dupont 2006)

- **Total complexity** (E. 2009)
  
  $O\left(|D|\right)$ — quasi-linear in the output size!
Implementation

- **Record (E. 2009) (with class invariants)**
  - $D = -2093236031$
  - $h = 100000$
  - Precision 264727 bits
  - 260000 s = 3 d CPU time
  - 5 GB

- **Software**
  - GNU MPC: complex floating point arithmetic in arbitrary precision with guaranteed rounding
    - Based on MPFR and GMP
    - LGPL
  - MPFRCX: polynomials with real (MPFR) and complex (MPC) coefficients
    - LGPL
  - cm: class polynomials and CM curves
    - GPL

http://www.multiprecision.org/
Further algorithms

- **$p$-adic lift**
  - Couveignes–Henocq 2002, Bröker 2006

- **Chinese remaindering**
  - Enumerate CM curves over $\mathbb{F}_p$, compute $H \mod p$
  - Lift to $\mathbb{Z}$ or directly to $\mathbb{Z}/P\mathbb{Z}$
  - Belding–Bröker–E.–Lauter 2008 following an idea by D. Bernstein, Sutherland 2009, E.–Sutherland 2010

- **Record (E.–Sutherland 2010)**
  - $D = -1 000 000 013 079 299$
  - $h = 10 034, 174$
  - $P \approx 2^{254}$
  - Precision 21 533 832 bits
  - 438 709 primes of $\leq 53$ bits
  - 200 d CPU time
  - Size $\mod P \approx 200$ MB
  - Size over $\mathbb{Z} \approx 2$ PB
Dupont 2006: One can evaluate $j$ at precision $n$ in time

$$O(\log n M(n)) = O^*(n).$$

Idea of the algorithm:
Newton iterations on a function built with the arithmetic-geometric mean (AGM)
Theta constants — definition

\[ a, b \in \frac{1}{2} \mathbb{Z}/\mathbb{Z}; \quad q = e^{\pi i \tau} \]

\[ \vartheta_{a,b}(\tau) = \sum_{n \in \mathbb{Z}} e^{\pi i ((n+a)\tau(n+a)+2(n+a)b)} = e^{2\pi i ab} \sum_{n \in \mathbb{Z}} (e^{2\pi i b})^n q^{(n+a)^2} \]

\[ \vartheta_{0,0}(\tau) = \sum_{n \in \mathbb{Z}} q^{n^2} = 1 + 2q + 2q^4 + 2q^9 + \ldots \]

\[ \vartheta_{0,\frac{1}{2}}(\tau) = \sum_{n \in \mathbb{Z}} (-1)^n q^{n^2} = 1 - 2q + 2q^4 - 2q^9 + \ldots \]

\[ \vartheta_{\frac{1}{2},0}(\tau) = \sum_{n \in \mathbb{Z}} q^{(2n+1)^2/4} = q^{1/4} (1 + 2q + 2q^3 + \ldots) \]

\[ \vartheta_{\frac{1}{2},\frac{1}{2}}(\mathbb{Z}) = 0 \]
Theta constants — duplication formulæ

\[
\vartheta^2_{0,0}(2\tau) = \frac{\vartheta^2_{0,0}(\tau) + \vartheta^2_{0,\frac{1}{2}}(\tau)}{2}
\]

\[
\vartheta^2_{0,\frac{1}{2}}(2\tau) = \sqrt{\vartheta^2_{0,0}(\tau)\vartheta^2_{0,\frac{1}{2}}(\tau)}
\]
AGM

\[
\vartheta_{0,0}^2(2\tau) = \frac{\vartheta_{0,0}^2(\tau) + \vartheta_{0,\frac{1}{2}}^2(\tau)}{2}
\]

\[
\vartheta_{0,\frac{1}{2}}^2(2\tau) = \sqrt{\vartheta_{0,0}^2(\tau)\vartheta_{0,\frac{1}{2}}^2(\tau)}
\]

AGM for \(a, b \in \mathbb{C}\)

- \(a_0 = a, \ b_0 = b\)
- \(a_{n+1} = \frac{a_n + b_n}{2}\)
- \(b_{n+1} = \sqrt{a_nb_n}\)

- converges quadratically towards a common limit \(\text{AGM}(a, b)\)

Evaluated in time \(O(\log n M(n))\) at precision \(n\).
Theta quotients

\[ \text{AGM}(a, b) = a \cdot \text{AGM}(1, b/a) =: a \cdot M(b/a) \]

- \[ k'(z) = \left( \frac{\vartheta_{0, \frac{1}{2}}(z)}{\vartheta_{0,0}(z)} \right)^2 \]
- \[ k(z) = \left( \frac{\vartheta_{\frac{1}{2},0}(z)}{\vartheta_{0,0}(z)} \right)^2 \]
- \[ k^2(z) + k'^2(z) = 1 \]
- \[ j = 256 \frac{(1-k'^2+k'^4)^3}{k'^4(1-k'^2)^2} \]
Newton iterations

- $M(k'(\tau)) = \frac{1}{\varphi_{0,0}^2(\tau)}$
- $M(k(\tau)) = M(k'(S\tau)) = \frac{1}{\varphi_{0,0}^2(S\tau)} = \frac{i}{\tau \varphi_{0,0}^2(\tau)}$
- $k^2(\tau) + k'^2(\tau) = 1$
- $f_\tau(x) = iM(x) - \tau M(\sqrt{1 - x^2})$
- $f_\tau(k'(\tau)) = 0$

\[
x_{n+1} \leftarrow x_n - \frac{f_\tau(x_n)}{f'_\tau(x_n)}
\]

converges quadratically towards $k'(\tau)$

Evaluated in time $O(\log n \ M(n))$ at precision $n$
1. **Elliptic curves**
   - Applications of elliptic curves
   - Complex multiplication theory
   - Algorithms

2. **Abelian surfaces**
   - Complex multiplication theory
   - Algorithms
   - Implementation and examples
Genus 2 curves and ppav of dimension 2

- $\mathcal{C} : Y^2 = X^5 + aX^3 + bX^2 + cX + d$ hyperelliptic curve of genus 2
- Jacobian is a principally polarised abelian surface (ppas)
- Moduli space of dimension 3 parameterised by Igusa invariants $i_1, i_2, i_3$
- Frobenius endomorphism gives cardinality of Jacobian over $\mathbb{F}_p$ → source of cryptographic curves
Endomorphism rings and period matrices

- \( \text{End} = \mathcal{O} \subseteq K = \mathbb{Q}[X]/(X^4 + AX^2 + B) \) with \( D = A^2 - 4B > 0 \)
- CM field of degree 4

\[
K = K_0 \left( \pm \sqrt{-\frac{A \pm \sqrt{D}}{2}} \right)
\]

\[
K_0 = \mathbb{Q}(\sqrt{D})
\]

- CM types \( \Phi = (\varphi_1, \varphi_2), \Phi' = (\varphi_1, \overline{\varphi_2}) \), embeddings: \( K \to \mathbb{C} \)
- \( (a, \xi) \) s.t. \( (a\overline{a}D)_{K/Q}^{-1} = (\xi), \varphi_1(\xi), \varphi_2(\xi) \in i\mathbb{R}_{>0} \) (polarisation)
- \( (a, \xi) \leadsto \tau = \begin{pmatrix} \tau_1 & \tau_2 \\ \tau_2 & \tau_3 \end{pmatrix} \) with \( \Im(\tau) \) positive definite (period matrix)
Theta constants

\[ a, b \in \left( \frac{1}{2} \mathbb{Z}/\mathbb{Z} \right)^2 \]

\[ \vartheta_{a,b}(\tau) = \sum_{n \in \mathbb{Z}^2} e^{\pi i ((n+a)^T \tau (n+a) + 2(n+a)^T b)} \]

10 non-zero theta constants

Siegel modular forms

Igusa invariants according to Streng 2010

\[ l_4 = \sum_{10 \, i} \vartheta_i^8 \]
\[ l_6 = \sum_{\text{certain } 60 \, i,j,k} \pm (\vartheta_i \vartheta_j \vartheta_k)^4 \]
\[ l_{10} = \prod_{10i} \vartheta_i^2 \]
\[ l_{12} = \sum \prod_{15 \, i, 6 \, i} \vartheta_i^4 \]

\[ i_1 = \frac{l_4 l_6}{l_{10}} \]
\[ i_2 = \frac{l_{12} p_4}{p_{10}^2} \]
\[ i_3 = \frac{p_4^5}{p_{10}^2} \]
Class fields (dihedral case)

\[ K^r(i_1(\tau)) = K^r(i_2(\tau)) = K^r(i_1(\tau)) \]
Elliptic curves
- Applications of elliptic curves
- Complex multiplication theory
- Algorithms

Abelian surfaces
- Complex multiplication theory
- Algorithms
- Implementation and examples
Algorithms

- **Complex analytic**
  - Spallek 1994
  - Weng 2001
  - Streng 2010

- **p-adic lift**

- **Chinese remaindering**
  - Eisenträger–Lauter 2005
  - Lauter–Robert 2012

- **Our contributions** to the complex-analytic algorithm
  - Quasi-linear evaluation of theta constants (following Dupont 2006)
    ⇒ quasi-linear computation of class polynomials (Streng 2010)
    ⇒ most efficient algorithm
  - Direct computation of irreducible factors, over $K_0^\ell$ instead of $\mathbb{Q}$
    (following Streng 2010)

**Software**

- Andreas Enge
- CM2
- LFANT 2015
Main algorithm (dihedral case)

- Let $h_0 = \#\text{Cl}(K_0)$, $h_1 = \#\text{Cl}(K)/h_0$
- Consider the two CM-types $\Phi$ and $\Phi'$, enumerate $\text{Cl}(K)$
- Compute

$$S(K, \Phi) = \{(a, \xi) : (a\bar{a}D_{K/Q})^{-1} = (\xi), \Phi(\xi) \in (i\mathbb{R}_{>0})^2\} / \sim$$

and $S(K, \Phi')$, where

$$(a, \xi) \sim (xa, (x\bar{x})^{-1}\xi)$$

- $\#S(K, \Phi) = \#S(K, \Phi') = h_1 \Rightarrow$ period matrices $\tau_i, \tau'_i$
- Evaluate the $\vartheta_{a,b}(\tau_i^{(\prime)})$ and deduce the $i_k(\tau_i^{(\prime)})$
Main algorithm (dihedral case)

- Compute the first class polynomial

\[ H_1(X) = \prod_{i=1}^{h_1} (X - i_1(\tau_i)) \prod_{i=1}^{h_1} (X - i_1(\tau_i')) \in \mathbb{Q}[X] \]

- Compute the Hecke representations of the algebraic numbers \( i_k(\tau_i) \) with respect to \( H_1 \):

The polynomial of degree \( h_1 - 1 \) such that

\[ i_k(\tau_i) = \frac{\hat{H}_k(i_1(\tau_i))}{\hat{H}_1'(i_1(\tau_i))} \]

(roughly Lagrange interpolation)
Borchardt sequences

\[ a_{n+1} = \frac{a_n + b_n + c_n + d_n}{4} \]
\[ b_{n+1} = \frac{\sqrt{a_n \sqrt{b_n}} + \sqrt{c_n \sqrt{d_n}}}{2} \]
\[ c_{n+1} = \frac{\sqrt{a_n \sqrt{c_n}} + \sqrt{b_n \sqrt{d_n}}}{2} \]
\[ d_{n+1} = \frac{\sqrt{a_n \sqrt{d_n}} + \sqrt{b_n \sqrt{c_n}}}{2} \]

Common limit: **Borchardt mean** \( B_2(a_0, b_0, c_0, d_0) \)

Related to duplication formulæ of four fundamental theta constants \( \vartheta_0, \ldots, \vartheta_3 \).
\( \tau \) from \( (\vartheta_j(\tau/2)/\vartheta_0(\tau/2))_{j=1,2,3} \)

- Compute \( (\vartheta_j^2(\tau)/\vartheta_0^2(\tau/2))_{j=0,1,2,3,4,6,8,9,12,15} \) (duplication)
- Compute \( B_2((\vartheta_j^2(\tau)/\vartheta_0^2(\tau/2))_{j=0,1,2,3} = \frac{1}{\vartheta_0^2(\tau/2)} \)
- Compute \( (\vartheta_j^2(\tau))_{j=0,1,2,3,4,6,8,9,12,15} \)
- Compute

\[
\begin{align*}
u_1 &= B_2((\vartheta_j^2(\tau))_{j=4,0,6,2} \\
u_3 &= B_2((\vartheta_j^2(\tau))_{j=8,9,0,1} \\
u_2 &= B_2((\vartheta_j^2(\tau))_{j=0,8,4,12} \\
\end{align*}
\]

- Return

\[
\tau_1 = \frac{i}{u_1}, \tau_3 = \frac{i}{u_3}, \tau_2 = \pm \sqrt{\frac{1}{u_2} + \tau_1 \tau_3}
\]
Streamline the computations

Replace 

\[ \frac{\partial f}{\partial \tau_i}(\tau) \]

by

\[ \frac{f(\tau + \varepsilon e_i) - f(\tau)}{\varepsilon} \]

(gain about 25%)
1. **Elliptic curves**
   - Applications of elliptic curves
   - Complex multiplication theory
   - Algorithms

2. **Abelian surfaces**
   - Complex multiplication theory
   - Algorithms
   - Implementation and examples
Implementation

- Number theoretic computations: $\mathcal{O}(K)$, (reduced) period matrices
  - Pari/GP
  - negligible effort

- Evaluation of theta and invariants
  - C
  - Libraries: GMP, MPFR, MPC
  - MPI for parallelisation

- Polynomial operations
  - MPFRCX
  - MPI for (partial) parallelisation

http://cmh.gforge.inria.fr/
Quasi-linear complexity

- required precision = coefficient size
- time per invariant = $\mathcal{O}(\text{precision})$
Quasi-linear complexity

- required precision = coefficient size
- time per invariant = $\mathcal{O}^\sim(\text{precision})$
- total time = $\mathcal{O}^\sim(\text{output size})$
Record example

- $K$ defined by $X^4 + 1357X^2 + 2122$, $D = 1832961$, $h_0 = 8$
- $\mathcal{C}(K) \simeq \mathbb{Z}/4402\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$
- PARI/GP: 4 min (reduction of period matrices)
- Precision: 7,536,929 bits
- Invariants:
  - Last Newton lift: $\approx 3000$ s per invariant ($\approx 1200$ second-to-last)
  - $\approx 2$ d wallclock time on 160 processors
- Polynomial operations (partially parallelised):
  - $\approx 1$ d wallclock time (40 processors, 1 TB memory)
- Algebraic coefficient recognition:
  - $\approx 2600$ s per coefficient
  - $\approx 10$ d wallclock time on 160 processors
- Size: 56 GB
- # primes in denominator: 3465
- Largest prime in denominator: 2,423,637,677
  - Bound: 54,004,867,207,824
Conclusion

- **Quasi-linear** algorithm for class polynomials in dimension 2
- Computation of invariants
  - efficient
  - arbitrarily parallel
- As can be expected: **Memory** becomes the bottleneck
- Better **parallelisation/distribution of polynomial operations** required
- **Quasi-linear LLL** in dimension 3 desirable
- Next steps:
  - better understand the denominators
  - smaller class invariants (work in progress with M. Streng)

http://cmh.gforge.inria.fr/

http://hal.archives-ouvertes.fr/hal-00823745/