Software presentation: theta constants and modular equations in genus 2

Jean Kieffer

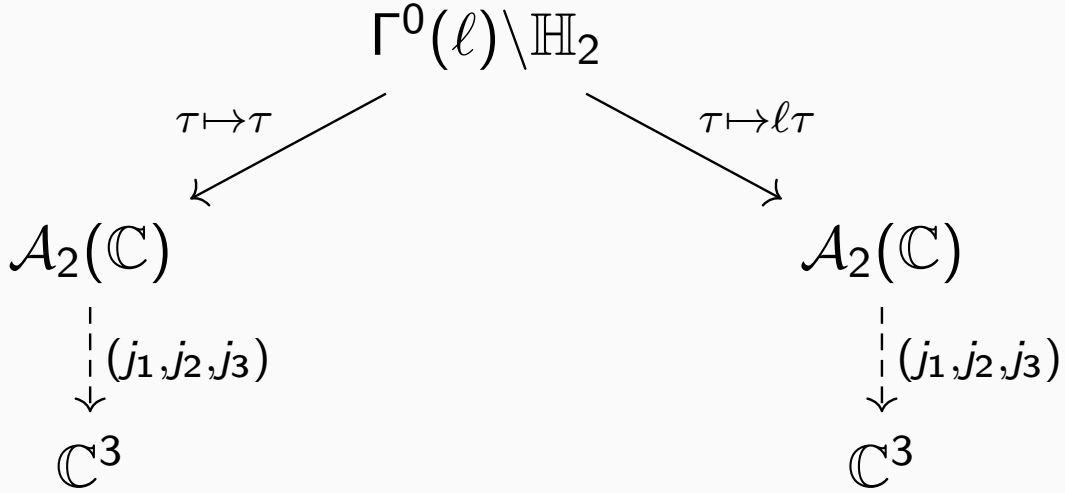
Friday 1 Oct. 2021

Simons Collaboration meeting

1. Mathematics

Siegel modular equations (1)

$\mathcal{A}_g(\mathbb{C}) = \Gamma(1)\backslash\mathbb{H}_g$, with $\Gamma(1) = \mathrm{Sp}_{2g}(\mathbb{Z})$.
Let ℓ be a prime, $g = 2$.



Siegel modular equations describe the image in $\mathbb{C}^3 \times \mathbb{C}^3$.

$$\Psi_{\ell,m} \in \mathbb{Q}(J_1, J_2, J_3)[X] \quad \text{for } 1 \leq m \leq 3.$$

Siegel modular equations (2)

Theorem

Let k be a field of characteristic prime to ℓ, and let A, B be p.p. abelian surfaces over k that are sufficiently generic. TFAE:

1. There exists an ℓ-isogeny $\phi : A \to B$ (i.e. $\ker\phi \subset A[\ell]$ is maximal isotropic, $\deg\phi = \ell^2$) defined over $\overline{k}$ (or even k);
2. $$\begin{cases} \Psi_{\ell,1}\big(j_1(A), j_2(A), j_3(A), j_1(B)\big) = 0, \\ j_2(B) = \Psi_{\ell,2}\big(j_1(A), j_2(A), j_3(A), j_1(B)\big)/D, \\ j_3(B) = \Psi_{\ell,3}\big(j_1(A), j_2(A), j_3(A), j_1(B)\big)/D, \end{cases}$$
   where $D = \dfrac{\partial\Psi_{\ell,1}}{\partial X}\big(j_1(A), j_2(A), j_3(A), j_1(B)\big)$.

The isogeny algorithm

If A, B satisfy the modular equations as above, and are sufficiently generic, then the data of

$$\frac{\partial \Psi_{\ell,m}}{\partial J_k}\big(j_1(A), j_2(A), j_3(A), j_1(B)\big) \quad \text{for } 1 \leq k, m \leq 3$$

is sufficient to recover an ℓ-isogeny $\phi : A \to B$ as a rational map, using the algorithm of [K., Page, Robert].

The Goal

Given a smooth genus 2 curve $\mathcal{C}/\mathbb{Q}$, evaluate

$$\Psi_{\ell,m}\big(j_1(\mathcal{C}), j_2(\mathcal{C}), j_3(\mathcal{C}), X\big) \in \mathbb{Q}[X], \text{ and}$$
$$\frac{\partial \Psi_{\ell,m}}{\partial J_k}\big(j_1(\mathcal{C}), j_2(\mathcal{C}), j_3(\mathcal{C}), X\big) \in \mathbb{Q}[X]$$

for $1 \leq k, m \leq 3$. Abbreviation: $j_k(\mathcal{C}) = j_k(\mathrm{Jac}(\mathcal{C}))$.

Theorem

Let $H = h(j_1, j_2, j_3)$ (logarithmic height). Then these polynomials have total size $O(\ell^6(H + \log \ell))$. Heuristically, they can be computed in quasi-linear time.

Remark: total size of $\Psi_{\ell,k} \in \mathbb{Q}(J_1, J_2, J_3, X)$ is $O(\ell^{15} \log \ell)$. Compare with elliptic modular polynomials: $O(\ell^3 \log \ell)$.

Hilbert case

Same game with Hilbert modular equations describing cyclic isogenies of degree ℓ between p.p.a. surfaces with real multiplication by $\mathbb{Z}_F$, where F is a real quadratic field. Size is $O_F(\ell^2(H + \log \ell))$.

Method

Given a smooth genus 2 curve $\mathcal{C}/\mathbb{Q}$, evaluate

$$\Psi_{\ell,m}\big(j_1(\mathcal{C}), j_2(\mathcal{C}), j_3(\mathcal{C}), X\big) \in \mathbb{Q}[X].$$

Use complex approximations:

1. Compute a period matrix τ of $\mathcal{C}$ to high precision.
2. List all period matrices $\gamma(\ell\tau)$ for $\gamma \in \Gamma^0(\ell)\backslash\Gamma(1)$, and compute Igusa invariants to high precision.
3. Reconstruct the polynomials using product trees.
4. Recognize rational (or often integer) coefficients.

Complexity awareness

All this has to be done in quasi-linear time in the precision, with the implied O constants depending only on H in a controlled manner. Dupont's algorithms (2006) are a key tool, but proper convergence is unfortunately heuristic.

For instance: I suspect that Magma's `AnalyticJacobian` is not quasi-linear time even for a single $\mathcal{C}$, and has no guarantee on uniformity, or even "correctness" of the output.

2. Software

C library

`hdme`: Higher-dimensional modular equations

|

`Arb`: Interval arithmetic in $\mathbb{R}$ and $\mathbb{C}$, polynomials and matrices

|

`Flint`: Arithmetic in $\mathbb{Z}$ and $\mathbb{Q}$, polynomials and matrices

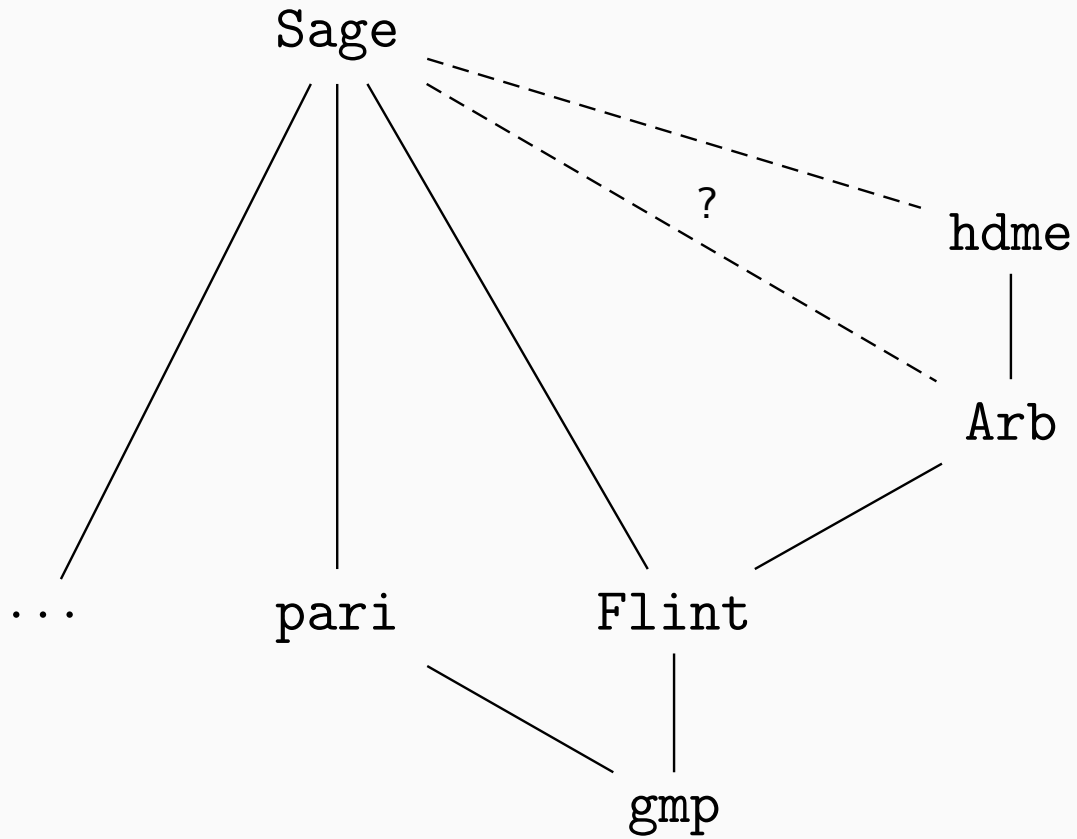
|

`gmp`: Operations in $\mathbb{Z}$

Why C?

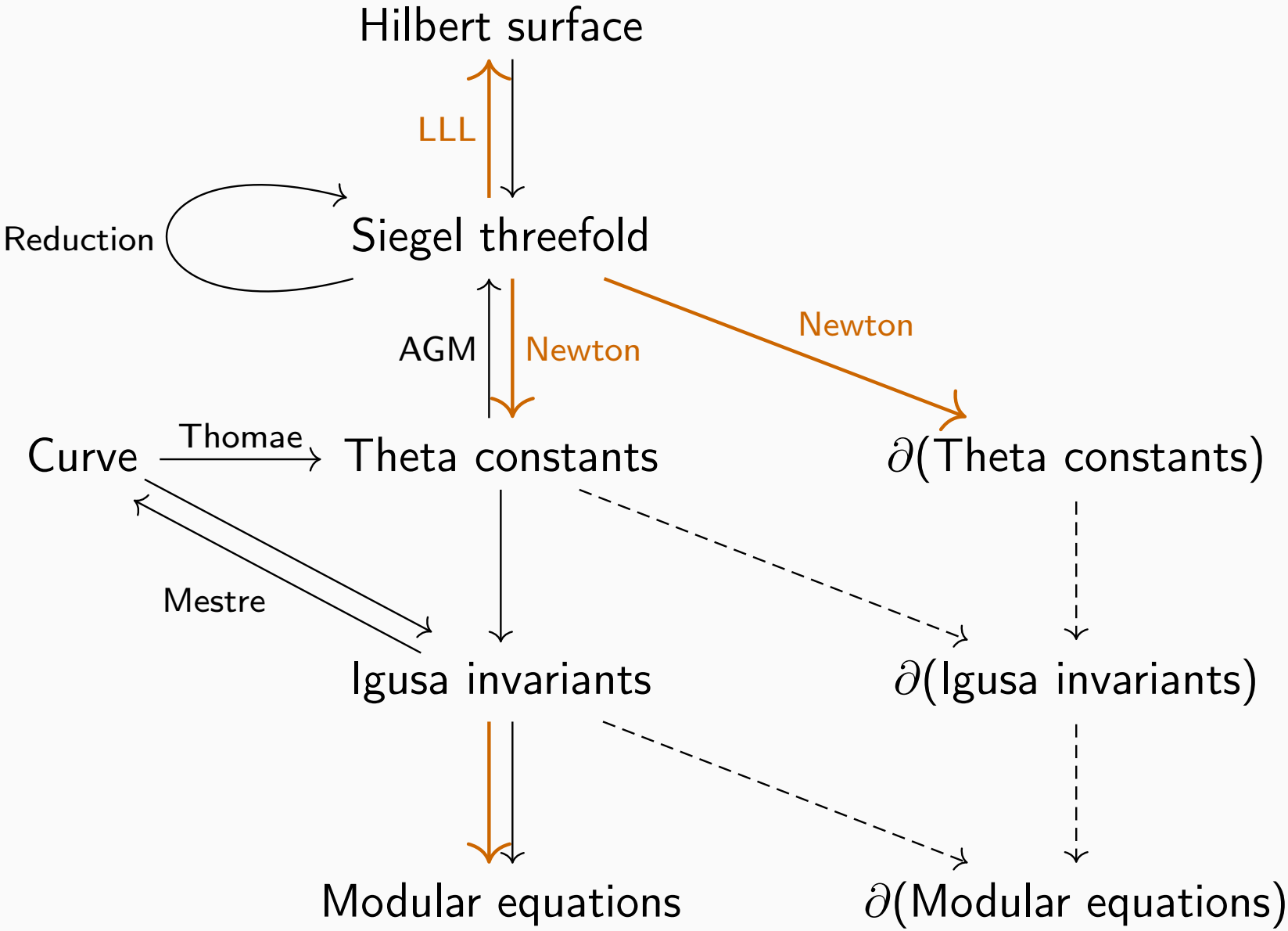
1. Modular equations are likely to be the bottleneck in many algorithms.
2. Arb offers control on errors during the computation, allowing provably correct results.
3. Fast evaluation of period matrices, Siegel modular forms and theta constants in genus 2 could make their way into Arb, for other applications.

Sage integration



I think the isogeny algorithm should be written in Sage inside the hyperelliptic curves module, or maybe in pari.

Contents


Hilbert surface
LLL
Reduction
Siegel threefold
AGM
Newton
Newton
Curve
Thomae
Theta constants
∂(Theta constants)
Mestre
Igusa invariants
∂(Igusa invariants)
Modular equations
∂(Modular equations)

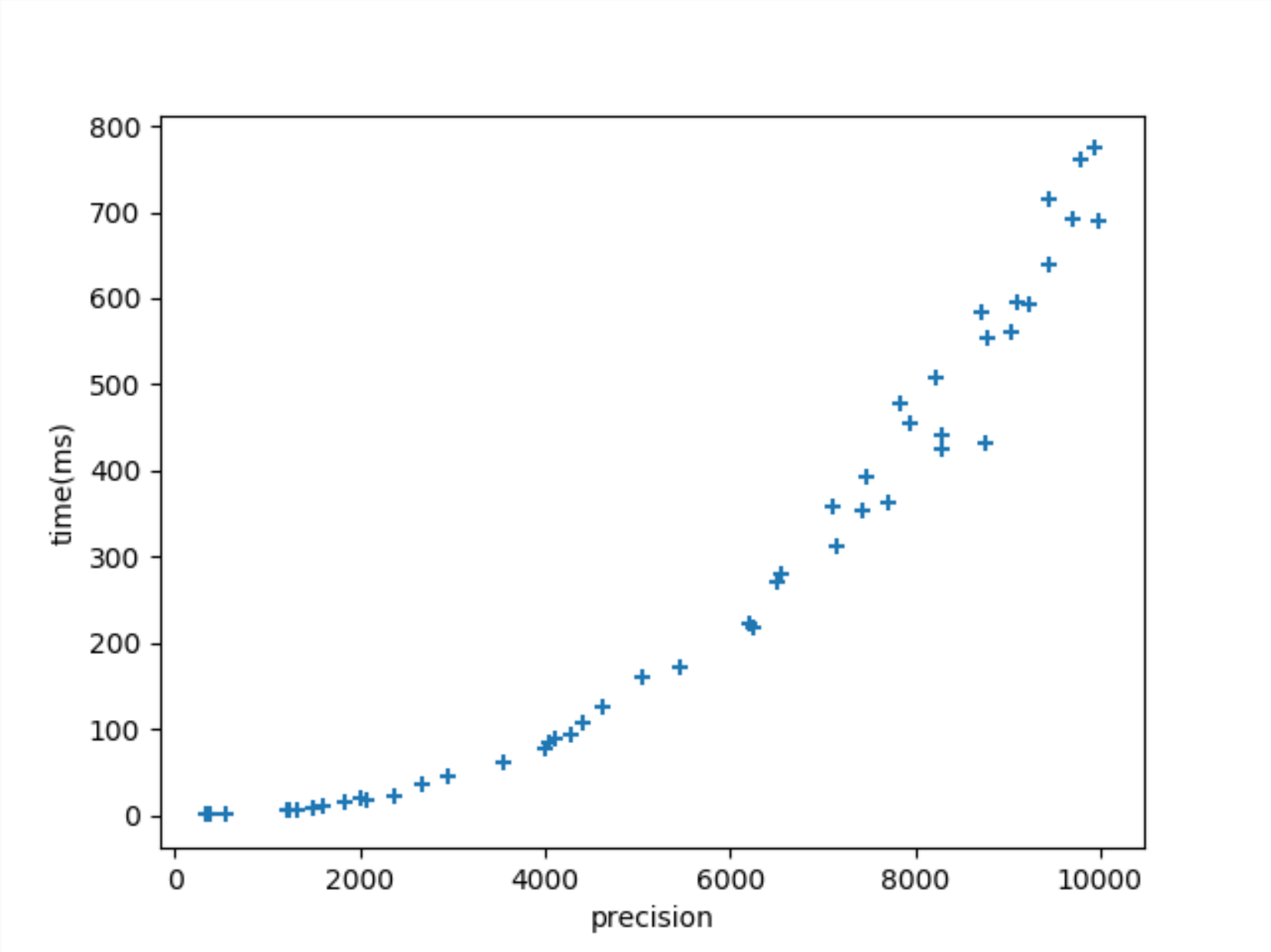
Examples (1)

```
int siegel_fundamental_domain(acb_mat_t w, sp2gz_t m,
const acb_mat_t z, const arb_t tol, slong prec);

int theta2_inverse(acb_mat_t tau, acb_srcptr th2,
slong prec);

int theta2_unif(acb_ptr th2, const acb_mat_t tau,
slong prec);

int mestre(acb_poly_t crv, acb_srcptr I, slong prec);

int tau_from_igusa(acb_mat_t tau, acb_srcptr I, slong
prec);
```

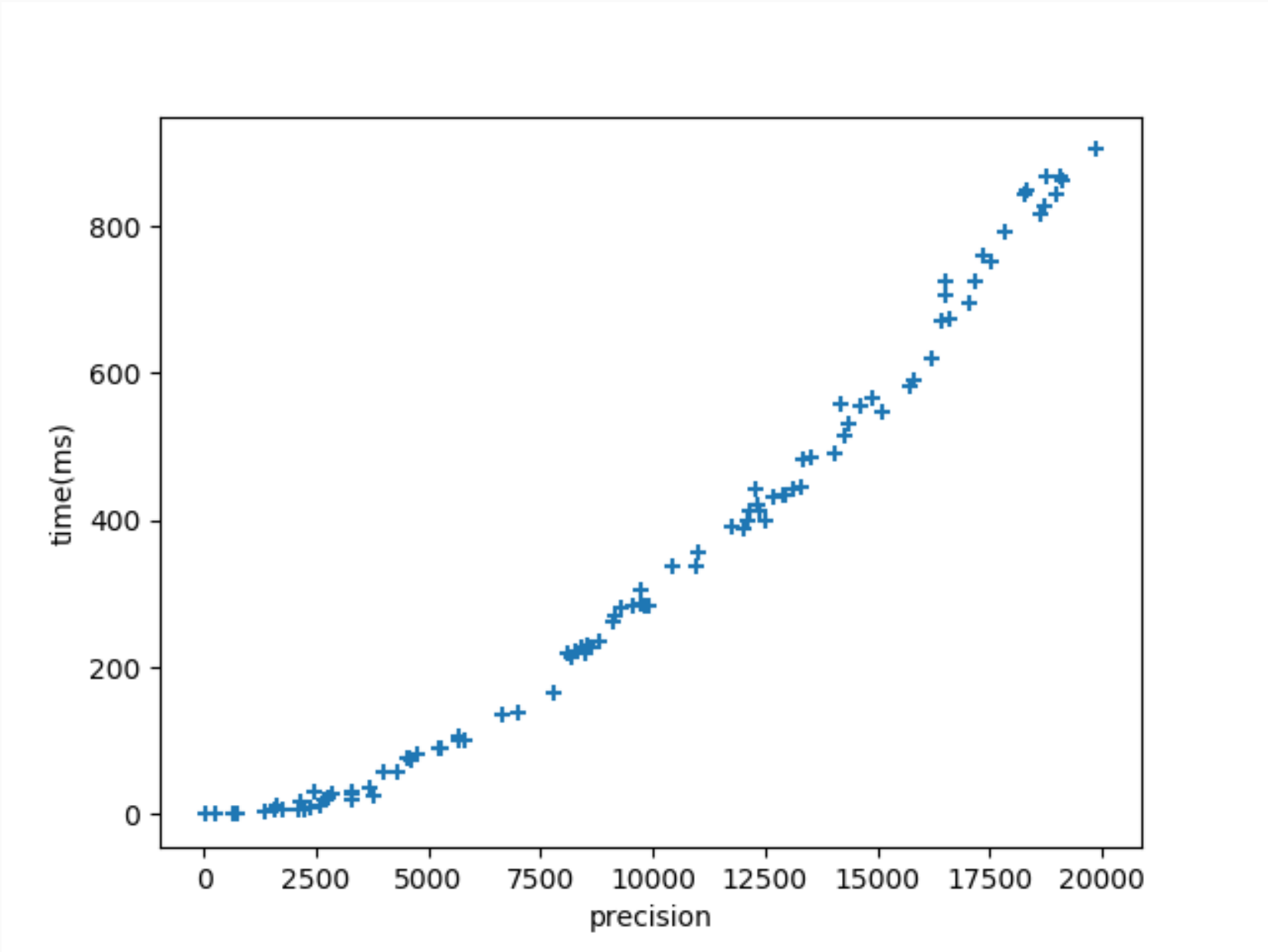
Examples (2)

```
int hilbert_inverse(acb_t t1, acb_t t2, sp2gz_t m,
const acb_mat_t tau, slong delta, slong prec);

int siegel_modeq_eval_Q(fmpz_poly_t num1, fmpz_poly_t
num2, fmpz_poly_t num3, fmpz_t den, fmpq* j, slong
ell);

int siegel_modeq_eval_Fp(fmpz_mod_poly_t pol1,
fmpz_mod_poly_t pol2, fmpz_mod_poly_t pol3, const
fmpz* j, slong ell, const fmpz_mod_ctx_t ctx);
```

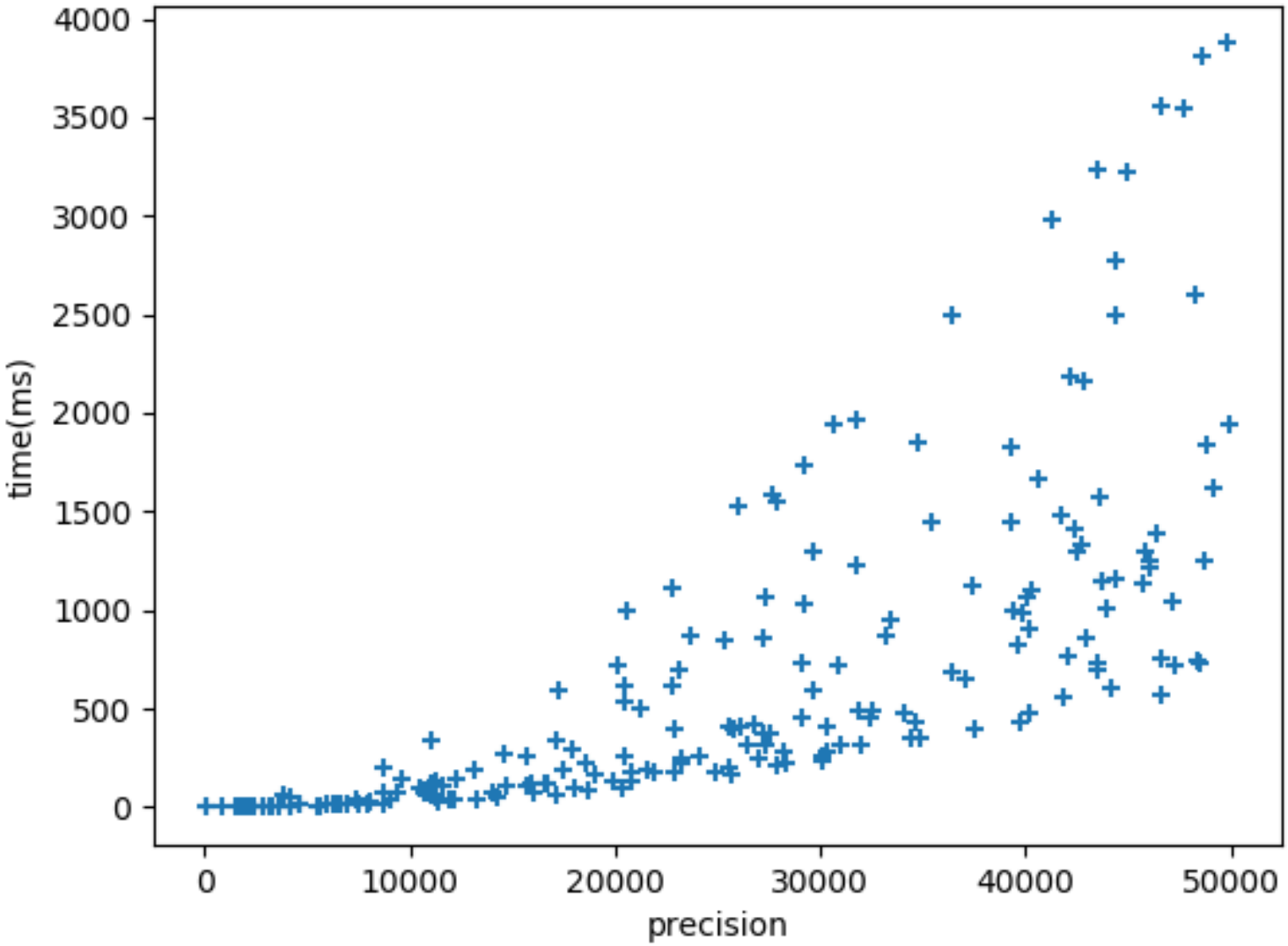
theta2_naive.c



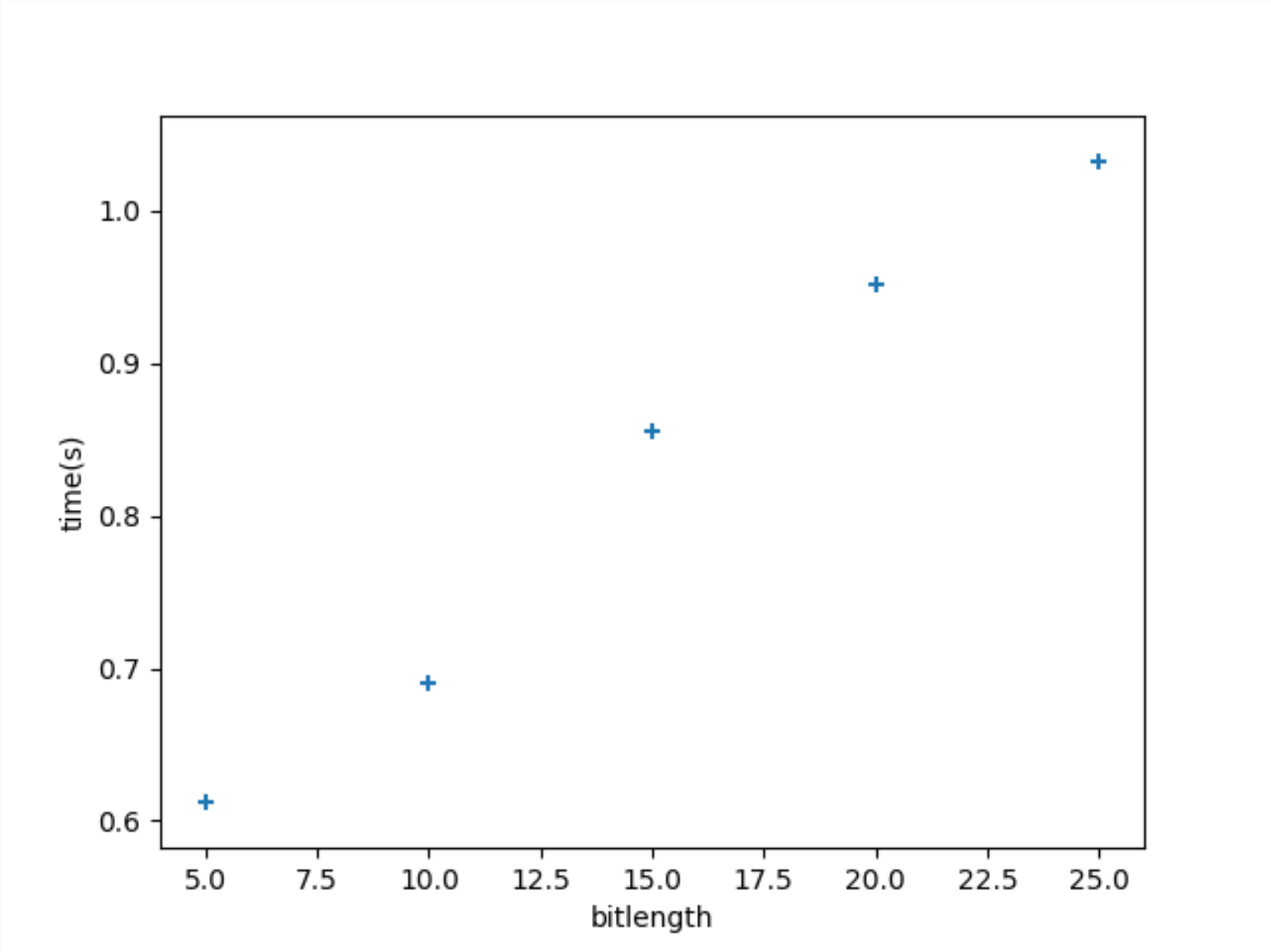
theta2_newton.c



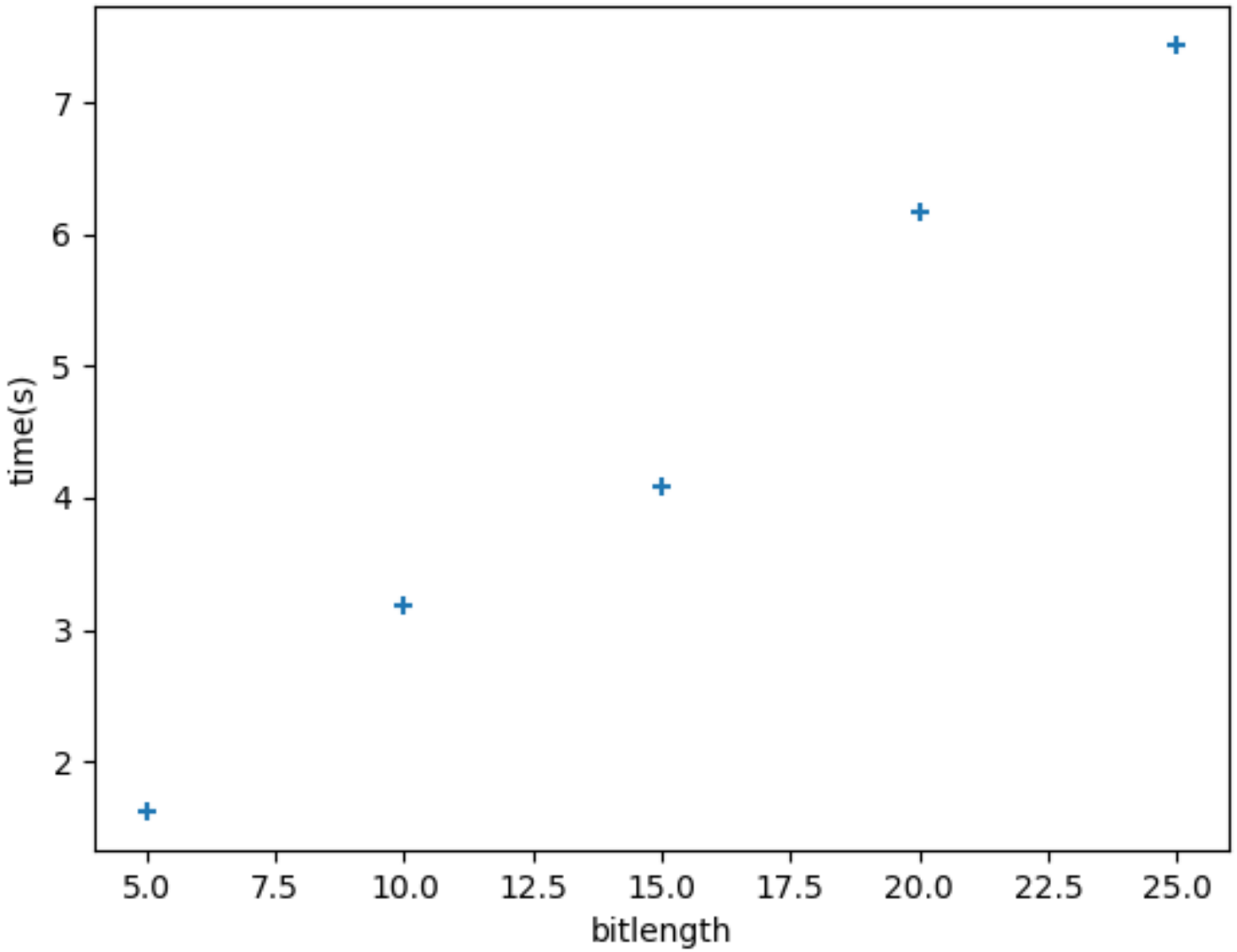
theta2_unif.c



Siegel modular equations, level 2



Siegel modular equations, level 3



Work in progress

Current code is tested and has no memory leaks.

1. Implement derivatives of modular equations.
2. Improve naming scheme.
3. Write more tests and benchmarks.
4. Write detailed documentation.
5. Choose license and make the code public.
6. Access from Sage, and implement the isogeny algorithm.
7. Try to "prove" correctness of heuristic steps (mathematical analysis, a posteriori certificates, use of Hilbert modular forms defined over $\mathbb{Z}$, ...)

Questions

Questions, comments, feature requests?