Mathematical Motivation for Quaternion Algebras in Sage

William Stein

August 25, 2014

Table of contents

1. Tamagawa numbers of modular abelian varieties
2. Explicit reduction of Kolyvagin classes
3. Hilbert modular forms and elliptic curves over $\mathbb{Q}(\sqrt{5})$
4. Much is missing

Quaternion algebras: More than just addition, subtraction, multiplication and division

```
Subject: Quaternions over number fields
Date: Fri, 22 Aug 2014 20:47:34 +0200 (CEST)
From: Waldek Hebisch <hebisch@math.uni.wroc.pl>
```

> *On the sage-devel you wrote that no open-source system can do quaternions over number fields. What do you mean by this? In FriCAS GeneralQuaternion domain implements quaternions over any commutative ring, in particular number fields. Do you mean some specific algorithms (FriCAS uses general algorithms and when something is very specific to quaternions than probably can not do this)? Or you need high efficiency?*
>
> *FriCAS implementation is quite small and presumably Sage could implement similar thing quite quickly, so I do not understand why you plan for ”next few years”.*

Quaternion algebras: More than just addition, subtraction, multiplication and division

> *"There are five elementary arithmetical operations: addition, subtraction, multiplication, division, and modular forms."* Eichler

arithmetic of quaternion algebras $\longleftrightarrow$ modular forms

My Ph.D. Thesis: BSD invariants of modular abelian varieties

- My Ph.D. thesis included an algorithm to compute (in many cases) the Tamagawa numbers c_p appearing in the BSD formula for modular abelian varieties A_f attached to newforms.
- When $p \mid\mid N$, key input to algorithm is a computation of an analogue of the modular degree involving the Hecke module on right ideal classes in an Eichler order of level $\frac{N}{p}$ in the quaternion algebra over $\mathbb{Q}$ ramified at p, ∞.
- When $N = p$, compute using Meste method of graphs.
- When $N \neq p$, David Kohel wrote Magma code to compute the same module as part of his Ph.D. work on endormorphism rings of elliptic curves.
- I started using/contributing to Magma as a result...

Demo: computing Tamagawa numbers in Sage

```
J = J0(65); J
```

Abelian variety J0(65) of dimension 5

```
D = J.decomposition(); D
```

[Simple abelian subvariety 65a(1,65) of dimension 1 of J0(65),
Simple abelian subvariety 65b(1,65) of dimension 2 of J0(65),
Simple abelian subvariety 65c(1,65) of dimension 2 of J0(65)]

```
A = D[1]; A
```

Simple abelian subvariety 65b(1,65) of dimension 2 of J0(65)

```
A.tamagawa_number(5) # quat algebras involved
```

7

```
A.tamagawa_number(13)
```

1

Kolyvagin classes

Let $E/\mathbb{Q}$ be an elliptic curve.

- Kolyvagin used Heegner points on modular curves to define elements of $H^1(K, E[p])$, for various number fields K.
- He used them to prove results about the BSD conjecture for curves with $r_{\mathrm{an}} \leq 1$.
- He conjectured that his classes were sometimes nonzero when $r_{\mathrm{an}} \geq 2$, but not one single case in which this was known.
- Can use quaternion algebras to explicitly compute reduction of classes, hence see they are nonzero.
- Christophe Cornut and Nike Vatsal also do this to prove "Mazur's conjecture".

Demo: Verifying instance of Kolyvagin's conjecture

The parameters below define a class $\tau_c \in H^1(\mathbb{Q}(\sqrt{-7}), E[3])$ for a rank 2 curve E.

We first try to verify Kolyvagin's conjecture for a rank 2 curve by working modulo 5, but we are unlucky with $c = 17$:

```
N = 389; D = -7; ell = 5; c = 17; q = 3
H = heegner_points(N).reduce_mod(ell)
E = EllipticCurve(’389a’)
V = H.modp_dual_elliptic_curve_factor(E, q, 5) # quat algs
k118 = H.kolyvagin_sigma_operator(D, c, 118)
k104 = H.kolyvagin_sigma_operator(D, c, 104)
[b.dot_product(k104.element().change_ring(GF(3))) for b in \
    V.basis()]
[b.dot_product(k118.element().change_ring(GF(3))) for b in \
    V.basis()]
```

```
[0, 0]
[0, 0]
```

Demo: Verifying Kolyvagin (continued)

Next we try again with $c = 41$ and this does work, in that we get something nonzero, when dotting with V, proving that $\tau_{41} \neq 0$:

```
c = 41
k118 = H.kolyvagin_sigma_operator(D, c, 118)
k104 = H.kolyvagin_sigma_operator(D, c, 104)
[b.dot_product(k118.element().change_ring(GF(3)\
   )) for b in V.basis()]
[b.dot_product(k104.element().change_ring(GF(3)\
   )) for b in V.basis()]
```

```
[1, 0]
[2, 0]
```

Kolyvagin's conjecture: follow up

- Big Theorem (Wei Zhang, 2014): Much of Kolyvagin's Conjecture was proved this year.
- Distribution: Based on numerical data at the Lopez Island Sage Days, Jared Weinstein and I proved precise distribution results (not published).
- Relevant code for above computations is in `src/sage/schemes/elliptic_curves/heegner.py`; there's various bits of code in there that should be moved to other places. *Good beginner project (for a number theorist).*

Enumerating elliptic curves over $F = \mathbb{Q}(\sqrt{5})$

- Problem: create tables like Cremona's over F.
- Paper: `http://wstein.org/papers/sqrt5/` – A Database of Elliptic Curves over $\mathbb{Q}(\sqrt{5})$—First Report. Enumerate a lot about elliptic curves over F with norm conductor ≤ 1831.
- Psage: Most code is in psage, which is a Python library. *Project: port code over to Sage, document, doctest, etc.*
- Dembele's algorithm: Psage includes blazingly fast implementation via computing with ideals in an Eichler order in the Hamilton quaternion algebra over F: gives Hilbert modular forms over F of weight (2,2).
- It took me about a month of hard work to go from what Dembele did to something that was actually fast based on it. Should have written a paper – but there's documented code. Uses a lot of C-level tricks, e.g., avoid malloc!

Non-demo: Hecke module with basis right ideal classes

- Uses **psage**: `http://purple.sagemath.org/`. Modularity *theorem* tells us enumerating eigenvectors here, same as elliptic curves over F. (Note: S. Pancratz and I spent a week in 2012 writing fast sparse linear algebra code so we could go to first curve of rank 4, than forgot all about it.)
- Psage is impossible to build against sage-6.x, since everything was dramatically reorganized in sage-6.0, which completely broke the psage build system. Also, I think the way imports work with Cython may have changed quite a bit.
- Moral: the only way to have code that uses Sage continue to work over time is to include it with Sage. Then it will not only to continue to work, but often gets better.
- Trust me: this code is much, much faster than the analoguous functionality in Magma. (Hundred times faster?)

Enumerating elliptic curves over other totally real fields

- Dembele and Voight of course know how to generalize most of the relevant quaternion algebras algorithms to all totally real fields: they did in papers, and implemented much in Magma.
- Likewise, regarding elliptic curves algorithms.
- However, making algorithms that are *blazingly fast* remains challenging. Focusing on specific fields can be very rewarding.

Quaternion Algebras in Sage: what's there?

- Fast basic arithmetic over number fields: Jon Bober and I did this at the Sage Days at Univ of Georgia.

```
... sage/algebras/quatalg$ ls
quaternion_algebra_element.pxd quaternion_algebra_element.pyx ...
```

- Eichler orders, Brandt modules (in some cases):

```
... sage/algebras/quatalg$ ls
quaternion_algebra_cython.pyx  quaternion_algebra.py  ...
... sage/modular/quatalg$ ls
all.py  brandt.py  __init__.py
```

- Aly Deines added a little bit of support for quaternion algebras over number fields. Daniel Smertnig fixed/added a few things over $\mathbb{Q}$ in 2012.
- Most experts use Magma for quaternion algebras, so what is in Sage is mostly (entirely?) motivated by my own research.

Wishlist for this week

- Get `sqrt5_fast.pyx` in psage to actually build.
- Move code from psage into Sage, satisfying all the coding/documentation constraints.
- Explore possibility of fast implementation for weights other than $(2,2)$.
- Explore possibility of fast implememntation for fields other than $\mathbb{Q}(\sqrt{5})$.
- Make a list of functionality for quaternion algebras (and Hilbert modular forms) that Magma has but Sage doesn't. How difficult would each thing be? Could Sage be better?
- Move some code from `heegner.py` to more sensible places in Sage (deprecation warnings).