A ROADMAP FOR QUATERNIONS IN SAGE

ALYSON DEINES, LASSINA DEMBELE, XAVIER GUITART, IAN KIMING, DAVID KOHEL, MARC MASDEU, MICHAEL NEURURER, AUREL PAGE, GUSTAVO RAMA, MATHIEU RAMBAUD, NADIM RUSTOM, JEROEN SIJSLING, NICOLAS SIROLLI, JAMES STANKEWICZ, WILLIAM STEIN, GONZALO TORNARIA, AND JOHN VOIGHT

This document contains a wish list for a robust quaternion package in `Sage`, as discussed at Sage Days 61 in Copenhagen in August 2014. The need for such a package is felt because the uses of quaternions are legion. To name a few:

- Through hyperbolic geometry, quaternion algebras give rise to generalizations of classical modular varieties, some of the simplest examples of which are Shimura curves. Having good quaternion functionality in `Sage` would make these objects more tangible.
- Quaternion algebras were essential in Wiles' proof of Fermat's last theorem and form an integral part of the Langlands program. This is most decidedly not merely a theoretical issue; many calculations involving Hilbert modular forms can be performed effectively by the clever use of quaternion algebras.
- In the study of counterexamples to the Hasse Principle, one needs to work with elements of Brauer groups and quaternion algebras are the most concrete examples of these.
- Considering quaternion algebras over the rationals leads to the study of genus 2 curves with many endomorphisms in characteristic 0 on the one hand and supersingular elliptic curves in characteristic p on the other.

The list of authors consists of the conference participants who took part in the discussion which led to the creation of this document. Special thanks go to Doctor Sijsling, who took notes during the discussion which slowly morphed into this document. This conference was supported by VILLUM FONDEN through the network for Experimental Mathematics in Number Theory, Operator Algebras, and Topology, and through the Department of Mathematics at the University of Copenhagen.

1. What would a robust implementation of quaternion algebras look like?

Before describing how to best improve the open source computation of quaternions, we will describe what kind of goals we are setting. In particular, below we describe "medium-big" dreams for a suite of quaternionic algorithms to be obtained over the next 2-5 years.

Following the famous Eichler quote that, "There are five basic arithmetic operations: addition, subtraction, multiplication, division, and modular forms," we organize our goals into five categories.

1.1. Quaternion algebras over fields.

To start with, we need methods to construct and manipulate quaternion algebras B over a field F. Later we will specify

F to be a number field, although there is certainly interest in function fields so that one can compute Drinfeld modular forms.

1.1.1. *Constructors.*

- Constructing B over a field of characteristic $\neq 2$ as $B = F \oplus Fi \oplus Fj \oplus Fk$, where $ij = -ji = k$, starting from the squares $a = i^2$ and $b = j^2$. The corresponding algebra is denoted by $\left(\frac{a,b}{F}\right)$. This is the preferable standard presentation of the algebra (in char$\neq 2$).
- Constructing B over a number field by specifying the ramifying places v of F, *i.e.*, those places v for which $B \otimes F_v \not\cong M_2(F_v)$.
- There should be a constructor which allows one to build a quaternion algebra over a field F by a quadratic field K and an element b of $F^\times$. We denote this algebra $\left(\frac{K,b}{F}\right)$. This is in some sense the correct way to think about characteristic 2 quaternion algebras although it seems likely that further optimization will be required.

1.1.2. *Basic arithmetic operations.* Although there are algorithms for addition, subtraction, multiplication, and division, it is important that these be optimized to be as fast as possible.

1.1.3. *Recognizing if a given associative algebra is a quaternion algebra.* Deciding whether or not a given associative algebra is central simple of dimension 4 over F, and if so, returning an isomorphism with a quaternion algebra $B = \left(\frac{a,b}{F}\right)$ in standard presentation.

1.1.4. *Identifying matrix rings and return isomorphisms.* Given a quaternion algebra in standard presentation $\left(\frac{a,b}{F}\right)$, we want to be able to efficiently decide if it is isomorphic to the matrix ring $M_2(F)$ and if so provide the desired isomorphism. The former is essentially the computation of Hilbert symbols. The latter is easy is one has a zerodivisor, and it is necessary and sufficient to find an F-rational point on a conic (see below).

1.1.5. *Isomorphism testing.* More generally, test for isomorphism between two different quaternion. At least over global fields, one can detect two algebras are isomorphic again by computing Hilbert symbols; exhibiting such an isomorphism can be achieved by finding a zero of the associated Albert form, a quadratic form in six variables.

1.1.6. *Hilbert symbols.* Given $B = \left(\frac{a,b}{F}\right)$, calculate the Hilbert symbol (a, b), which is 1 if and only if the conic $ax^2 + by^2 = z^2$ has an F-rational point and -1 otherwise. The algebra B is isomorphic to $M_2(F)$ if and only if $(a, b) = 1$.

1.1.7. `pMatrixRing` *for Fields.* Return an isomorphism $B \otimes F_v \to M_2(F_v)$ if B is split at v and an isomorphism between B and the non-split quaternion algebra over F_v otherwise. It remains to decide which standard presentation to take for the latter algebra, specifically which unit and uniformizer to take.

1.1.8. *Norm equations/Embeddings of quadratic fields.* The algebra $B = \left(\frac{a,b}{F}\right)$ splits if and only if the element b is a norm for the extension $F(a)|F$.

A quadratic extension K of F embeds into B if and only if $B \otimes_F K$ is split. Given K, determine whether or not K embeds into B by using norm equations, and give the embedding explicitly if it does.

1.1.9. *Relations with conics and quadratic spaces.* Given B, the 3-dimensional F-vector space B_0 of elements of trace 0 is equipped with the quadratic form coming from the reduced norm on B. Equating the norm to zero gives rise to a conic Q in the corresponding 2-dimensional projective space over F.

Another way to phrase the key identity for norm equations and field embeddings is that the algebra B splits if and only if the associated conic Q admits a point over F. Implement all links of this kind between quaternion algebras over F and conics and/or quadratic spaces over F.

1.2. **Quaternion orders.** Given an algebra B over a number field F, one can consider the lattices of B, *i.e.*, the projective $\mathbb{Z}_F$-modules of rank 4 in B, as well as the orders of B, *i.e.*, those lattices of B that are also subrings of B. This section considers the orders, while the desired functionality for lattices is described in the next section.

An Eichler order $\mathcal{O}$ is the intersection of two maximal orders $\mathcal{O}_1, \mathcal{O}_2$; its level is the $\mathbb{Z}_F$ ideal $\mathfrak{N}$ such that $\mathcal{O}_1/\mathcal{O} \cong (\mathbb{Z}_F/\mathfrak{N})$ as $\mathbb{Z}_F$-modules. In what follows, we do not assume any of the orders involved to be Eichler unless explicit mention to the contrary is made.

It is often worthwhile to consider more general lattices and orders, *i.e.*, over small base rings R than $\mathbb{Z}_F$, as well as S-integral orders $\mathcal{O}$.

1.2.1. *Constructors.* If we want quaternion orders in more generality than class number one number fields, we are going to have to put some work into creating a more robust suite of algorithms for computing with modules over Dedekind domains. Implement the following methods to construct orders:

- Constructing an Eichler order $\mathcal{O}$ from the discriminant of the algebra B and the level N of $\mathcal{O}$. More generally, construct the orders of given index in a maximal order. Currently maximal orders are implemented over $\mathbb{Z}$, but this functionality should exist in all cases.
- Construct the unique order $\mathcal{O}$ that is free of rank 4 over $\mathbb{Z}_F$ from the discriminants D_1, D_2, D_3 of the elements i, j, k such that $\mathcal{O} = \mathbb{Z}_F \oplus \mathbb{Z}_F i \oplus \mathbb{Z}_F j \oplus \mathbb{Z}_F k$.
- Construct the unique order corresponding to a ternary quadratic form over $\mathbb{Z}_F$ via the Clifford Functor.

1.2.2. *Optimize order representation.* Optimize the representation of the order by using LLL over $\mathbb{Z}$ or $\mathbb{Z}_F$ to find "smaller" isomorphic representations, as in the case of number fields.

1.2.3. *Diminishing and enlarging.* Given two orders $\mathcal{O}, \mathcal{O}'$, calculate their intersection. Given an order $\mathcal{O}$ and an element x, determine the smallest order containing both if it exists.

1.2.4. *Containing orders.* Given an order $\mathcal{O}$, construct an order $\mathcal{O}'$ containing $\mathcal{O}$ that is maximal at a given (cofinite) set of places S of F. More generally, construct all orders containing $\mathcal{O}$.

1.2.5. *Suborders.* Given a finite bound and an order $\mathcal{O}$, determine all suborders of $\mathcal{O}$ of index up to that bound (e.g., using Sirolli's algorithm).

1.2.6. *Embeddings and Embedding numbers of quadratic orders.* Given an order R in a quadratic extension K of F, return the embeddings (or only the optimal ones if so desired) of R into the order $\mathcal{O}$. Determine the corresponding CM points after the implementation of the corresponding hyperbolic geometry; see also the section on S-units below.

1.2.7. *Bases.* Write a function `HasBasis` that returns `true` if and only if $\mathcal{O}$ is a free module over $\mathbb{Z}_F$. More generally, write functions that return a pseudo-basis or a $\mathbb{Z}$-basis of $\mathcal{O}$. As much computation depends on finding good pseudobases quickly, this is a major goal with implications far beyond quaternions.

1.2.8. *Conjugacy.* Given two orders $\mathcal{O}$ and $\mathcal{O}'$, determine if $\mathcal{O}$ and $\mathcal{O}'$ are conjugate. In general, knowing that two orders are isomorphic, find a conjugation that effects this isomorphism after Skolem-Noether.

1.2.9. *Eichler orders as intersections.* Given an Eichler order $\mathcal{O}$, find two maximal orders $\mathcal{O}_1$ and $\mathcal{O}_2$ such that $\mathcal{O} = \mathcal{O}_1 \cap \mathcal{O}_2$.

1.2.10. *Eichler invariants.* Given an order $\mathcal{O}$ and a prime $\mathfrak{p}$ of $\mathbb{Z}_F$, let J be the Jacobson radical of $\mathcal{O}/\mathfrak{p}\mathcal{O}$. Determine the Eichler invariant of $\mathcal{O}$. This invariant is 0 if J has dimension 3; otherwise, it is 1 if $(\mathcal{O}/\mathfrak{p}\mathcal{O})/J$ is a sum of two copies of $\mathbb{Z}_F/\mathfrak{p}$, and it is -1 if $(\mathcal{O}/\mathfrak{p}\mathcal{O})/J$ is the quadratic extension of $\mathbb{Z}_F/\mathfrak{p}$.

1.2.11. *Mass formula.* Given a quaternion order $\mathcal{O}$ in a rational definite quaternion algebra B of discriminant D, it is often much easier to write a weighted sum over the projective right ideal classes than the actual size of the class group. In particular, if $\mathcal{O}(I)$ denote the left order of a right ideal I and $w(I) = \#\mathcal{O}(I)/2$ then $\sum 1/w(I)$ is called the Eichler mass of $\mathcal{O}$. In particular, if $\mathcal{O}$ is maximal then the mass of $\mathcal{O}$ is simply $\phi(D)/12$. This equality is known as the Eichler mass formula and its ease of computation makes it especially useful. Generalizations to other settings become easy once Eichler invariants are implemented. [KV10]

1.2.12. *Relations with quadratic spaces.* As in the field case, return the trace 0 lattice with the corresponding quadratic form. This is already implemented over the integers.

1.2.13. *Elements of norm.* Given a bound B and an order $\mathcal{O}$, find the elements of norm up to B (up to the units of $\mathcal{O}$). This functionality is extremely important for the computation of Hecke operators. As such this should be made as efficient as possible. There is a crucial distinction here also between the definite case, which works with quadratic forms, and the harder indefinite case which works by finding explicit generators for ideals.

1.2.14. *pMatrixRing.* Given a place $\mathfrak{p}$ of $\mathbb{Z}_F$ where B splits, determine an isomorphism $B \to M_2(F_\mathfrak{p})$ that sends the order $\mathcal{O}$ to a standard form at that prime, such as the usual upper triangular representation in the case of Eichler orders. [Voi13]

1.2.15. *Predicates.* Once orders are implemented, it should be both easy and highly beneficial to have some of the basic predicates implemented. Some examples follow.

- `IsMaximal`
- `IsEichler`
- `IsHereditary`
- `IsPrimitive`
- `IsBass`
- `IsGorenstein`
- `IsSelective`

1.3. **Ideals and lattices.** Given a lattice I and an order $\mathcal{O}$ in B, we say that I is a left (resp. right) $\mathcal{O}$-ideal if $\mathcal{O}I = I$ (resp. $I\mathcal{O} = I$). Every lattice is an ideal for some order $\mathcal{O}$; the biggest $\mathcal{O}$ for which $\mathcal{O}I = I$ (resp. $I\mathcal{O} = I$) is called the left (resp. right) order of I.

Given an order $\mathcal{O}$, one often uses the set $\mathrm{Cl}(\mathcal{O})$ of (left or right) ideals of $\mathcal{O}$ up to principal ideals; two ideals I and I' are in the same left (resp. right) ideal class if and only if $I' = Ib$ (resp. $I' = bI$) for some $b \in B$.

1.3.1. *Basic operations.* Given two ideals I_1, I_2 and an element b of B, determine the join $I_1 + I_2$, the meet $I_1 \cap I_2$, the colon lattice $(I_1 : I_2)$, the quotient I_1/I_2 if it exists, the scalar multiplication bI_1, the conjugate ideal $\overline{I}$, and the inverse ideal I^{-1}. Determine the left and right order of I_1.

1.3.2. *Morphisms.* Implement a class of morphisms for lattices.

1.3.3. *Isomorphism testing.* Determine whether or not two lattices are isomorphic, *i.e.*, related by a conjugation, by using ternary quadratic forms if necessary. Algorithms are completely different in the definite vs. indefinite case [Pag14].

1.3.4. *Principality.* Determine whether a given $\mathcal{O}$-ideal I is (narrowly) principal, and if so, determine a generator. Combined with local splitting, this is especially relevant for ideals of Eichler orders, whose *localizations* are always principal.

1.3.5. *Ideal class enumeration.* Given an order $\mathcal{O}$, enumerate its (left, right or two-sided) ideal classes. [KV10]

1.3.6. *Pic of $\mathcal{O}$.* It would be beneficial to have implemented the stably free class group $K_0(\mathcal{O})$ of an order $\mathcal{O}$.

1.3.7. *Ideal class representatives.* Given an ideal class and a prime ℓ not dividing the discriminant, determine a representative ideal I whose norm is a power of ℓ [KLPT14].

1.3.8. *General generators.* For general ideals I, find a 2-generating set, consisting of a norm in $\mathbb{Z}_F$ and an element of B generating away from the primes dividing the specified norm.

1.3.9. *Bases.* Given an ideal I, determine a (pseudo-)basis, as well as an LLL-reduced basis for efficiency purposes.

1.3.10. *Subideals.* Given an $\mathcal{O}$-ideal I and a suborder $\mathcal{O}'$ of $\mathcal{O}$, determine those ideals I' of $\mathcal{O}'$ such that $\mathcal{O}I' = I$. More generally, if we have a class of morphisms implemented, there should be functionality for pushforward and pullback of ideals. Nicolás Sirolli has some code for this.

1.3.11. *Elements of norm.* Given a bound B and an ideal I, find the elements of norm up to B (up to the units of I).

1.3.12. *Actions.* Given an order $\mathcal{O}$ over a base R, determine the action of $\mathrm{Cl}(R)$ on $\mathrm{Cl}(\mathcal{O})$.

1.3.13. *Theta series.* Given an ideal I, compute its theta series.

1.4. **S-Units.** Given a cofinite set of places S, one is interested in determining the structures of the finitely generated groups associated with an S-integral order $\mathcal{O}$. Think of the groups $\mathcal{O}_1^\times$ (elements of norm 1), $\mathcal{O}_+^\times/\mathbb{Z}_F^\times$ (elements of totally positive norm modulo scalars), as well as the elements $N(\mathcal{O})_+^\times/F^\times$ in the normalizer of the latter group.

1.4.1. *Presentation.* Given a unit group U, return a presentation in terms of generators and relations. Perhaps this could be done through a package like GAP.

1.4.2. *Word problem.* Given a unit group U with a presentation and an element u of U, determine an expression for u as a word in the generators. (Using the fundamental domain mentioned below for example.)

1.4.3. *Hyperbolic geometry.* Implement the Poincaré disk / upper half plane and its geometry, as well as the corresponding generalizations in higher dimension. There is a library CGAL that is very nice and relevant to this, and has a compatible license.

1.4.4. *Signature.* Given a unit group U acting on an hyperbolic space H, determine the signature of the orbifold H/U. Related (also with the mass formula above): Determine the covolume of U.

1.4.5. *Fundamental domains.* Given a unit group U acting on an hyperbolic space H, determine a fundamental domain. (This gets harder and harder as the degree of the field and the number of split places increases.)

1.4.6. *Bruhat-Tits theory.* The $\mathfrak{p}$-adic version of the complex analytic theory above; develop functionality for actions on the Bruhat-Tits tree and consider the corresponding Mumford curves *etc.*. [FM14]

1.5. **Modular forms.** What it is in the end all about; applications to modular forms and systems of Hecke eigenvalues.

1.5.1. *Brandt modules.* Give an implementation of (abstract) Brandt modules of arbitrary character and weight, after Kohel *et al.*. In particular, an implementation of Brandt modules with a compatible representation theory package

1.5.2. *Atkin-Lehner operators.* Determine the action of the Atkin-Lehner operators $N(\mathcal{O})_+^\times F^\times/\mathcal{O}_+^\times F^\times$ on Brandt modules. After choosing a basis, this action is represented by Brandt matrices.

1.5.3. *Actions on cohomology.* Generalize the previous item to compute actions of Atkin-Lehner and Hecke operators on the cohomologies associated with quaternionic groups.

1.5.4. *Relations with arithmetic geometry.* We have now come full circle as applications to arithmetic geometry are a strong motivation for "thinking quaternionically."

2. How to develop this functionality: Logistics

We first recount a discussion, conducted about the logistics of `Sage`; in other words, who will code what needs to be coded? This is a discussion which needs to take place because tenured staff do not have many opportunities to implement algorithms. There is much teaching pressure, and usually one resorts to sabbaticals or periods of unpaid leave.

This is an especially salient point for the implementation of quaternions because there is quite a lot of quaternion arithmetic implemented in MAGMA, and reimplementing a piece of software is often more onerous than figuring out how things go for the first time. The obvious choice is to have young people do so, in addition to coding they can learn about the surrounding mathematical objects in a very deep way.

In enlisting young mathematicians for this work however, some care must be taken. Graduate students also often have teaching commitments and of course their first priority is to complete their thesis. It was noted that coding projects assigned to graduate students should be consonant with their research goals. However, there is only a certain extent to which this can be true. Efficiency is a key factor here and there is usually significant time to be invested in somewhat unpleasant running time issues.

A key example which was brought up many times was the contrast of Brandt Modules versus that of Hermite normal form for modules over a Dedekind domain. Although the former is mathematically substantial and a student would learn much from coding it, any code robust enough to work over a number field with nontrivial class group would require a working copy of the latter. Many pieces of code are like the former and all those would benefit from having an indispensable piece of code like HNF working as efficiently as possible.

Although that graduate students will work hard for the sake of getting better at computational algebraic number theory, there needs to be some enticement to spend time on coding the essentials instead of on teaching or other graduate student requirements. There is great benefit to having a group of graduate students all working together, although there is the question of where the money to replace teaching with coding would come from.

The ideal that our group came upon was that of a "`Sage` development fellowship" wherein 5-10 graduate students could have their teaching bought out in order to code and 3-5 postdoctoral fellows could be kept onboard to maintain a general view and infrastructure. An alternative is also suggested by the recent Google "Summer of Code," in which five students were sponsored to write code over the summer. Graduate students in need of summer support would be well-served by such a program, although it should be noted that there will be competition from industry during the summer for mathematics graduate students who can code and stipends should reflect that.

3. How to develop this functionality: Structure

We have thus far discussed what needs to be implemented and who should do it. We now come to the crucial question of how this should be done. In particular, we've singled out the following projects as being most important to be done in the next few months.

3.1. The next few months to a year.

(1) Basic operations on ideals and lattices: this includes the creation of a `QuaternionLattice` class, which has been begun at this conference. In particular, until a fast implementation is given for Hermite normal forms, the concentration is on working over rings of integers of number fields with trivial class group. In this case, we can use existing infrastructure for torsion-free modules over Principal Ideal Domains. For the future we can install `NotImplementedError`s.

(2) `pMatrixRing`: Given an order, it is vital to able to test its isomorphism class. This project was frequently brought up as an excellent candidate for something that a graduate student could learn a lot from coding.

(3) Isomorphism testing of ideals and lattice: This is another piece of basic functionality which needs to be built into the `QuaternionLattice` package. This will allow us to enumerate a complete set of representatives for the class group of an order $\mathcal{O}$.

(4) Optimized Class Representatives: When we can compute a complete list of inequivalent ideals for an order $\mathcal{O}$, the next crucial step is to find optimized representatives for the isomorphism classes.
   - All of the above can be done over rings of integers of class number one fields. It is even conceivable that over such fields, there may be a speed increase over other implementations. It is possible that one might be able to implement something which works for free modules over Dedekind domains, but at that point we're just putting off a very important enhancement.

(5) Fast Hermite Normal Form: This is necessary to anything going forward over number fields. It is also worth noting here that there is an implementation of this in PARI which could be wrapped, although the code is somewhat old. That said, there did seem to be some interest on the part of the PARI team in improving that part of the code.

References

[FM14] Cameron Franc and Marc Masdeu, *Computing fundamental domains for the Bruhat–Tits tree for* $\mathrm{GL}_2(\mathbf{Q}_p)$*, p-adic automorphic forms, and the canonical embedding of Shimura curves*, LMS J. Comput. Math. **17** (2014), no. 1, 1–23.

[KLPT14] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol, *On the quaternion* ℓ*-isogeny path problem*, LMS J. Comput. Math. **17** (2014), no. A, 418–432.

[KV10] Markus Kirschmer and John Voight, *Algorithmic enumeration of ideal classes for quaternion orders*, SIAM J. Comput. **39** (2010), no. 5, 1714–1747.

[Pag14] A. Page, *An algorithm for the principal ideal problem in indefinite quaternion algebras*, LMS J. Comput. Math. **17** (2014), no. A, 366–384.

[Voi13] John Voight, *Identifying the matrix ring: algorithms for quaternion algebras and quadratic forms*, Quadratic and higher degree forms, Dev. Math., vol. 31, Springer, New York, 2013, pp. 255–298.