Subfields and Abelian overfields

A. Page

INRIA/Université de Bordeaux

21/01/2020
Institut Fourier

Plan

This tutorial:

- construction of subfields of a number field
- construction of abelian extensions of a number field

These are old functionalities but we made a number of changes to them.

If you want to record the commands we will type during the tutorial:

```
? \l subsupfields.log
```

Subfields

We compute the subfields of a number field with the function `nfsubfields`.

```
? pol1 = y^8 - y^6 + 2*y^2 + 1;
? #nfsubfields(pol1)
% = 6
```

This number field has 6 subfields.

```
? #nfsubfields(pol1,4)
% = 3
```

Three of them have degree 4 over $\mathbb{Q}$.

Subfields and embeddings

For each subfield, the function gives a defining polynomial and an element of the large field defining the embedding.

```
? sub1 = nfsubfields(pol1,4)
% = [[y^4 - ... , 2*y^7 - ...], [y^4 - ...,
 -2*y^7 + ...], [y^4 - ..., -y^2 + 1]]
? a = y^2+y;
? minpoly(Mod(a,sub1[1][1]))
% = x^4 + 2*x^3 + 76*x^2 + 60*x + 12
```

We can compute the image of *a* in the large field with `subst`.

```
? minpoly(Mod(subst(a,y,sub1[1][2]),pol1))
% = x^4 + 2*x^3 + 76*x^2 + 60*x + 12
```

Subfields

We can also use an `nfinit` structure as input.

```
? nf1 = nfinit(pol1);
? #nfsubfields(nf1,2)
% = 1
```

Algorithms for subfields

Depending on the situation, we use various algorithms to compute subfields of $K = \mathbb{Q}[X]/P(X)$.

1. Galois theory (Allombert);
2. A combinatorial algorithm (Klüners);
3. A factorisation based algorithm (van Hoeij – Klüners – Novocin).

1. is always faster when available, 3. is polynomial-time, and 2. is exponential in the worst case but it is often fast.

In 3., we need the factorisation of P over K.

Subfields: providing the factorisation

We can provide the factorisation to the function. This forces the use of Algorithm 3. and saves the recomputation of the factorisation.

```
? fa1 = nffactor(pol1, subst(pol1,y,x));
? sub1b = nfsubfields([pol1,fa1],4)
% = [[y^4 + ..., -y^5 + ...], [y^4 + ..., -y^2],
 [y^4 + ..., -y^3 + y]]
```

We can check that we obtained the same subfields with `nfisisom`.

```
? nfisisom(sub1[1][1],sub1b[1][1])
% = [-1/2*y^3 - 1/2*y^2 - 3/2*y - 1/2,
 1/2*y^3 - 1/2*y^2 + 3/2*y - 1/2]
```

Subfields: providing the factorisation

There is no canonical ordering for the subfields, so they may end up being permuted.

```
? matrix(#sub1,#sub1b,i,j,
    nfisisom(sub1[i][1],sub1b[j][1])!=0)
% =
[1 0 0]

[0 0 1]

[0 1 0]
```

Maximal subfields

We can restrict to the enumeration of maximal subfields with the function `nfsubfieldsmax`.

```
? {pol2 = x^16 - 4*x^15 + 34*x^14 - 102*x^13 +
 620*x^12 - 1542*x^11 + 7436*x^10 - 14962*x^9 +
 67815*x^8 - 111634*x^7 + 409898*x^6 - 504000*x^5
 + 1459447*x^4 - 1224212*x^3 + 3769899*x^2 -
 1828918*x + 6914293};
? sub2 = nfsubfieldsmax(pol2);
? apply(a -> poldegree(a[1]), sub2)
% = [4, 8, 8, 8]
```

They do not always have the same degree.

Maximal subfields: providing the factorisation

This uses a variant of Algorithm 3., and we can also provide the factorisation.

```
? fa2 = nffactor(pol2, subst(pol2,x,t));
  *** incorrect priority: variable t >= x
```

Watch out for the priority of variables!

```
? t = varhigher("t");
? fa2 = nffactor(pol2, subst(pol2,x,t));
? nfsubfieldsmax([pol2,fa2]) == sub2
% = 1
```

Descending further

We can then compute subfields of the maximal subfields, etc.

```
? {pol3 = y^12 + 6*y^10 - 10*y^9 + 36*y^8 - 60*y^7
+ 276*y^6 - 720*y^5 + 1776*y^4 - 2360*y^3 +
2160*y^2 - 1200*y + 400};
? sub3 = nfsubfieldsmax(pol3);
? apply(a -> poldegree(a[1]), sub3)
% = [4, 6]
? sub3b = nfsubfieldsmax(sub3[1][1])
% = [[y^2 - ...ugly...]
```

We can simplify the models with `polredbest`.

```
? polredbest(sub3b[1][1])
% = y^2 - y - 1
```

CM fields

Recall that a number field K is called *CM* (complex multiplication) if it is a totally imaginary quadratic extension of a totally real field.

In this case, it admits an automorphism of order 2 which induces complex conjugation on every embedding of K into $\mathbb{C}$; this automorphism is called the CM involution or the complex conjugation on K.

Maximal CM subfield

We can also compute the maximal CM subfield (if it exists).

```
? nfsubfieldscm(pol1)
% = [y^2 + 3, 2*y^6 - 4*y^4 + 2*y^2 + 3]
? sub2b = nfsubfieldscm([pol2,fa2])
% = [x^4 + ...*x^2 + ..., ...]
```

The computed model always satisfies that $x \mapsto -x$ is the CM involution.

In `polredbest`, we can keep track of the change of variable with an optional flag $= 1$.

```
? polredbest(sub2b[1],1)
% = [x^4 - 2*x^3 - 11*x^2 + 12*x + 57, ...]
? polredbest(substpol(sub2b[1],x^2,x))
% = x^2 - 7
```

Abelian extensions of $\mathbb{Q}$

Recall that every Abelian extension of $\mathbb{Q}$ is contained in a cyclotomic field (Kronecker–Weber).

`polsubcyclo`(n, d) computes every subfield of $\mathbb{Q}(\zeta_n)$ of degree d.

```
? polsubcyclo(23,11)
% = x^11 + x^10 - 10*x^9 - 9*x^8 + 36*x^7 + 28*x^6
 - 56*x^5 - 35*x^4 + 35*x^3 + 15*x^2 - 6*x - 1
```

`galoissubcyclo` computes the subfield fixed by a given subgroup of $(\mathbb{Z}/n\mathbb{Z})^\times$.

```
? #polsubcyclo(60,8)
% = 7
? galoissubcyclo(60,-1)
% = x^8 - 7*x^6 + 14*x^4 - 8*x^2 + 1
```

Abelian extensions of $\mathbb{Q}$

We compute the structure and generators of $(\mathbb{Z}/n\mathbb{Z})^\times$ with `znstar`.

```
? G = znstar(7*13*19)
% = [1296, [36, 6, 6], [Mod(743, 1729), Mod(248, 17
```

We can describe the subgroup in terms of those generators.

```
? H = mathnfmodid([1,0;-1,1;0,-1],3);
? galoissubcyclo(G,H)
% = x^3 + x^2 - 576*x - 64
? nfdiscfactors(%)
% = [2989441, [7, 2; 13, 2; 19, 2]]
```

Abelian extensions of number fields

In general, the Abelian extensions of a number field K are the subfields of its ray class fields, whose Galois groups are canonically isomorphic to the ray class groups $\mathcal{C}\ell_K(\mathfrak{m})$. (Class field theory)

The special case $\mathfrak{m} = (1)$ is the Hilbert class field.

Transcendental methods

In some cases we can use transcendental methods to compute ray class fields.

Hilbert and ray class fields of quadratic fields:

```
? quadhilbert(-23)
% = x^3 - x^2 + 1
? quadray(-7,8)
% = x^8 + Mod(-4*y + 4, y^2 - y + 2)*x^7 + ...
```

Assuming Stark's conjectures, ray class fields of totally real fields:

```
? bnrstark(bnrinit(bnfinit(y^3-y^2-41*y+104),1))
% = x^9 + ...
```

Kummer theory method

In all cases we can use Kummer theory. This can be costly since we need to compute the class group and units of $K(\zeta_p)$ to compute extensions of degree p of K, and towers of such for general Abelian extensions.

The function `rnfkummer` is now obsolete; use the more general `bnrclassfield` instead, which we will present now.

Hilbert class field

```
? pol4 = y^2-y+1007
% = y^2 - y + 1007
? bnf = bnfinit(pol4); bnf.cyc
% = [3, 3]
```

The class group is isomorphic to $\mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}$.

```
? ext4 = bnrclassfield(bnf)
% = [x^3 - 15*x + (-1204*y + 602), x^3 + ...]
```

By default, the class field is expressed as the compositum of two degree 3 extensions. We can compute a single defining polynomial with `nfcompositum`.

```
? nfcompositum(bnf,ext4[1],ext4[2],2)
% = x^9 + ...
```

Hilbert class field

We can directly ask for a single relative defining polynomial with an optional flag $= 1$.

```
? bnrclassfield(bnf,,1)
% = x^9 + 18*x^7 + ...
```

We can also ask for a single absolute defining polynomial with an optional flag $= 2$.

```
? bnrclassfield(bnf,,2)
% = x^18 + 36*x^16 + 4860*x^14 + ...
```

Ray class groups

We compute general ray class groups with `bnrinit`.

```
? pr = idealprimedec(bnf,13)[1];
? bnr = bnrinit(bnf,pr); bnr.cyc
% = [18, 3]
```

This ray class group is isomorphic to $\mathbb{Z}/18\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}$. We can compute the discriminant of the corresponding extension in advance with `bnrdisc`.

```
? [deg,r1,D] = bnrdisc(bnr);
? deg
%59 = 108
? D
% = 62583356628008526 8...18199167302475256851237
```

Ray class fields

```
? ext2 = bnrclassfield(bnr)
% = [x^2 + (y + 34), x^3 + ..., x^9 + ...]
```

Again, the ray class field is expressed as a compositum of several extensions.

We can simplify the relative defining polynomials with `rnfpolredbest`.

```
? apply(P -> lift(rnfpolredbest(bnf,P)), ext2)
% = [x^2 + (y + 34), x^3 - 24*x + (2*y - 1),
x^9 - x^8 + (-y - 5)*x^7 + ... + (-262*y + 10515)]
```

Ray class fields

Again, we can ask for an absolute defining polynomial.

```
? ext2b = bnrclassfield(bnr,,2)
% = x^108 + 24*x^107 + 229*x^106 - 128*x^105 - ...
```

We can check that the discriminant is correct with `nfdisc`.

```
? nfdisc([ext2b,1000]) == D
% = 1
```

Note that this is much more expensive than with `bnrdisc`, and we needed to help `nfdisc` by forcing it to use a lazy factorisation.

General class fields

In general we describe the desired Abelian extension as the subfield of a ray class field fixed by a subgroup of $\mathcal{C}\ell_K(\mathfrak{m})$.

```
? pr2 = idealprimedec(bnf,2)[1];
? bnr2 = bnrinit(bnf,[pr,1;pr2,3]); bnr2.cyc
% = [36, 12, 6]
? H2 = [2,1,1;0,2,0;0,0,1]
% =
[2 1 1]
[0 2 0]
[0 0 1]
? bnrclassfield(bnr2,H2)
% = [x^4 + 78*x^2 + (-92*y + 1396)]
```

Shortcut for describing the subgroup

We can use the shortcut `bnrclassfield(bnr,n)` to denote the subgroup $n \cdot \mathcal{C}\ell_K(\mathfrak{m})$.

```
? ext3 = bnrclassfield(bnr2,3)
% = [x^3 - 15*x + ..., x^3 + ..., x^3 + ...]
```

This is the maximal elementary Abelian 3-subextension.

```
? ext3 = bnrclassfield(bnr2,9)
% = [x^3 + ..., x^3 + ..., x^9 + ...]
```

This is the maximal Abelian subextension with exponent dividing 9.

Without the explicit field

Computing a defining polynomial with `bnrclassfield` can be time-consuming, so it is better to compute the relevant information without constructing the field, if possible.

We already saw the use of `bnrdisc`; we can also compute splitting information without the explicit field.

```
? pr313 = idealprimedec(bnf,313)[1];
? bnrisprincipal(bnr2,pr313,0)
% = [0, 0, 0]~
```

The Frobenius at $\mathfrak{p}_{313}$ is trivial: this prime splits completely in the degree $36 \cdot 12 \cdot 6 = 2592$ extension (which we did not compute).

Modulus with infinite places

If the base field has real places, we can specify the modulus at infinity by providing a list of 0 or 1 of length the number of real embeddings.

```
? bnf2 = bnfinit(y^2-217);
? bnf2.cyc
% = []
? bnrinit(bnf2,1).cyc
% = []
? bnr3 = bnrinit(bnf2,[1,[1,1]]); bnr3.cyc
% = [2]
```

The field $\mathbb{Q}(\sqrt{217})$ has narrow class number 2.

A narrow Hilbert class field

We check that the class field has the expected properties:

```
? [deg,r1,D] = bnrdisc(bnr3);
? [deg,r1]
% = [4, 0]
? D
% = 47089
? bnrclassfield(bnr3)
% = [x^2 + (-260952*y + 3844063)]
? pol5 = bnrclassfield(bnr3,,2)
% = x^4 + 7688126*x^2 + 1
? polsturm(pol5)
% = 0
? nfdisc(pol5) == D
% = 1
```

Questions ?

Have fun!