

# Computing field degrees of radical extensions

Willem Jan Palenstijn  
Universiteit Leiden



Intercity number theory seminar  
Nijmegen, February 24, 2006

(updated 27 Februari 2006)

## Field degrees

**Definition.** For  $S \subset \mathbf{Q}^*$ ,  $S \neq \emptyset$ , write

$$\sqrt[n]{S} = \{x \in \overline{\mathbf{Q}}^* : x^n \in S\}.$$

**Theorem.** There is a polynomial-time algorithm with

**Input:**

$n \in \mathbf{Z}_{>1}$ ,  $S \subset \mathbf{Q}^*$ ,  $S$  finite,  $S \neq \emptyset$ .

**Output:**

The degree of  $\mathbf{Q}(\sqrt[n]{S})$  over  $\mathbf{Q}(\zeta_n)$ .

## Cyclotomic part

Why not the degree of  $\mathbf{Q}(\sqrt[n]{S})$  over  $\mathbf{Q}$  ?

$$[\mathbf{Q}(\zeta_n) : \mathbf{Q}] = \varphi(n)$$

Given  $n$  and  $\varphi(n)$ , there is a fast probabilistic algorithm to factor  $n$  by finding non-trivial square roots of 1 in  $(\mathbf{Z}/n\mathbf{Z})^*$ .

## Abelian groups

$n \in \mathbf{Z}_{>1}$   
 $B \in \overline{\mathbf{Q}}^*$  with  $B/\mathbf{Q}^*$  finite of exponent  $n$ .

Write  $w = \#\mathbf{Q}^*[n]$ , so  $w \in \{1, 2\}$ .

For  $k \in \mathbf{Z}_{>1}$  write  $\mu_k = \overline{\mathbf{Q}}^*[k]$ .

Assume  $\mu_{nw} \subset B$ .

### **Theorem.**

$\text{Gal}(\mathbf{Q}(B)/\mathbf{Q})$  is a normal subgroup of  $\text{Aut}_{\mathbf{Q}^*}(B)$  with abelian cokernel  $E(B)$ , the *entanglement group*.  $E(B)$  can be explicitly computed.

# Automorphisms

Input of algorithm:

$n \in \mathbf{Z}_{>1}$ ,  $S \subset \mathbf{Q}^*$ ,  $S$  finite,  $S \neq \emptyset$ .

Define:

$$B' = \langle \mathbf{Q}^*, \sqrt[n]{S} \rangle$$

$$B = \langle B', \zeta_{nw} \rangle$$

Steps of algorithm:

- Compute  $\#\text{Aut}_{\mathbf{Q}^*}(B)$  (up to  $\varphi(n)$ ).
- Compute  $\#E(B)$ .
- Compute  $[\mathbf{Q}(B) : \mathbf{Q}(B')]$ .

First step: compute

$$\frac{\#\text{Aut}_{\mathbf{Q}^*}(B)}{\varphi(nw)} = \#\text{Aut}_{\langle \mathbf{Q}^*, \zeta_{nw} \rangle}(B).$$

# Factoring

Basic idea: consider  $\langle S \rangle$  as a sub- $\mathbf{Z}$ -module of  $\mathbf{Q}^*$ , and apply linear algebra.

Problem:

can't factor  $n$  and the elements of  $S$  into primes.

# Factoring

Solution:

factor  $n$  and the elements of  $S$  into *coprimes*.

**Definition.** Let  $T \subset \mathbf{Z}_{>0}$ . A set of positive integers  $\mathcal{P}$  is a *coprime base* for  $T$  if the elements of  $\mathcal{P}$  are pairwise coprime and  $T$  is contained in the monoid generated by  $\mathcal{P}$ .

**Theorem** (D. J. Bernstein).

There is an algorithm with essentially linear runtime that, given a finite set  $T \subset \mathbf{Z}_{>0}$ , computes a finite coprime base for  $T$ .

## Coprime bases

$$S \subset \mathbf{Z}_{>0}$$

**Definition.** The *closure* of  $S$  is the smallest  $T \subset \mathbf{Z}_{>0}$  satisfying:

- $S \subset T, 1 \in T$ ;
- If  $a, b \in T$ , then  $ab \in T$ ;
- If  $ab, b \in T$ , then  $a \in T$ ;
- If  $a, b \in T$ , then  $\gcd(a, b) \in T$ .

**Definition.** If  $T$  is the closure of  $S$ , the set of minimal elements of  $T \setminus \{1\}$  is the *natural coprime base* of  $S$ , written  $\text{cb}(S)$ .

## Definitions

**Definition.** Let  $S$  be a set of integers. Then  $\text{prod}(S)$  is the product of all elements of  $S$ .

**Definition.** Let  $x$  be a positive integer. Then  $\text{rad}(x)$  is the product of all primes dividing  $x$ .

## Building block

Subalgorithm (“ $ppo(a, b)$ ”):

**Input:**  $a, b \in \mathbf{Z}_{>0}$ .

**Output:** Largest divisor  $r$  of  $a$  that is coprime to  $b$ ; and  $s = a/r$ .

Note:  $\text{rad}(s) \mid b$ .

Step 1: set  $s = (a, b)$  and  $r = \frac{a}{(a, b)}$ .

Now  $s$  contains the right primes, but  $r$  is not necessarily coprime to  $b$ .

Repeat: set  $g = (s, r)$ ; set  $s = sg$ ,  $r = r/g$ , until  $g = 1$ .

This “gathers” all primes in  $s$  that are also in  $r$  into  $s$ .

Output  $r$  and  $s$ .

## Example

$$a = 12, b = 2.$$

First:  $s = (12, 2) = 2$  and  $r = 6$ .

Repeating:

$g = (2, 6) = 2$ , so  $s = 2 \cdot 2 = 4$ ,  $r = 6/2 = 3$ .

$g = (4, 3) = 1$ , so we are done.

## Building block

Subalgorithm (“ppg( $a, b$ )”):

**Input:**  $a, b \in \mathbf{Z}_{>0}$ .

**Output:**  $s = \text{ppg}(a, b)$ , which is defined by:

for all primes  $p$ ,

$\text{ord}_p(s) = \text{ord}_p(a)$  if  $\text{ord}_p(a) > \text{ord}_p(b)$ ;

$\text{ord}_p(s) = 0$  otherwise.

Step 1: set  $s = a / (a, b)$  and  $r = (a, b)$

Now  $s$  contains the right primes. As before, “gather” all of those primes remaining in  $r$  into  $s$ .

## Two-element sets

$x, y \in \mathbf{Z}_{>0}$ .

Computing  $\text{cb}(\{x, y\})$ :

Start by computing the largest divisor  $r$  of  $x$  coprime to  $y$ . Output  $r$  and replace  $x$  by  $x/r$ .

All primes in  $x$  now also occur in  $y$ .

We will separate them by comparing orders of primes (with “ppg”):

Recall:  $\text{ppg}(x, y)$  is the part of  $x$  consisting of primes which have a higher power in  $x$  than in  $y$ .

## Example

$$x = 2 \cdot 3^2 \cdot 5^6, \quad y = 2 \cdot 3 \cdot 5$$

Step 1:

$$g_1 = \gcd(x, y) = 2 \cdot 3 \cdot 5.$$

$$h_1 = \text{ppg}(x, y) = 3^2 \cdot 5^6; \quad x/h_1 = 2.$$

$\gcd(x/h_1, y) = 2$  gives the prime 2.

Step 2:

$$g_2 = \gcd(h_1, g_1^2) = 3^2 \cdot 5^2.$$

$$h_2 = \text{ppg}(h_1, g_1^2) = 5^6; \quad h_1/h_2 = 3^2.$$

$\gcd(h_1/h_2, y) = 3$  gives the prime 3.

Step 3:

$$g_3 = \gcd(h_2, g_2^2) = 5^4.$$

$$h_3 = \text{ppg}(h_2, g_2^2) = 5^6; \quad h_2/h_3 = 1.$$

Step 4:

$$g_4 = \gcd(h_3, g_3^2) = 5^6.$$

$$h_4 = \text{ppg}(h_3, g_3^2) = 1; \quad h_3/h_4 = 5^6.$$

$\gcd(h_3/h_4, y) = 5$  gives the prime 5.

## Extending a coprime base

$\mathcal{P}$  a coprime set,  $x \in \mathbf{Z}_{>0}$ .

Computing  $\text{cb}(\mathcal{P} \cup \{x\})$ :

Compute  $\text{prod}(\mathcal{P})$  and the largest divisor  $r$  of  $x$  coprime to  $\text{prod}(\mathcal{P})$ .

If  $r \neq 1$ , output  $r$ .

Replace  $x$  by  $x/r$ .

Each prime factor of  $x$  now occurs in some element of  $\mathcal{P}$ .

## Extending a coprime base, cont.

We split  $x$  into factors:

$$x = \prod_{p \in \mathcal{P}} x_p,$$

with  $\text{rad}(x_p) \mid p$ .

This can be done by splitting  $\mathcal{P}$  into two halves, and doing this step for each half separately by recursing. The factor of  $x$  (not) to use for each half can be computed with the “ppo” subalgorithm.

For each factor  $x_p$ , compute  $\text{cb}(x_p, p)$  and output it.

## Extending coprime bases, example

$$\mathcal{P} = \{4, 15\}, x = 66.$$

$\text{prod}(\mathcal{P}) = 60$ ; largest divisor  $r$  of 66 coprime to 60 is 11.

The remainder is  $66/11 = 6$ .

Split  $\mathcal{P}$  into  $\mathcal{P}_1 = \{4\}$ ,  $\mathcal{P}_2 = \{15\}$ .

Part of  $x$  relevant for  $\mathcal{P}_1$  is 2.

$$\text{cb}(4, 2) = \{2\}.$$

Part of  $x$  relevant for  $\mathcal{P}_2$  is 3.

$$\text{cb}(3, 15) = \{3, 5\}.$$

Result:  $\{11, 2, 3, 5\}$ .

## Merging coprime bases

$\mathcal{P}$  and  $\mathcal{Q}$  coprime sets.

Computing  $\text{cb}(\mathcal{P} \cup \mathcal{Q})$ :

Idea 1: add  $\mathcal{Q}$  to  $\mathcal{P}$  one element at a time.

However,  $\mathcal{Q}$  can have too many elements to have time for this.

Idea 2: first replace  $\mathcal{Q}$  by a smaller set  $X$  for which  $\text{cb}(X) = \mathcal{Q}$ .

## Constructing a smaller set

How to construct such a “small” set  $X$  with  $\text{cb}(X) = \mathcal{Q}$  ?

Example:  $\mathcal{Q} = \{q_0, q_1, \dots, q_7\}$ .

Write the indices  $0, \dots, 7$  in binary.

Take  $X$  the following 6 products of  $q_i$ 's:

bit 0 is 0:	$q_0$	$q_2$	$q_4$	$q_6$	
bit 0 is 1:	$q_1$	$q_3$	$q_5$	$q_7$	
bit 1 is 0:	$q_0$	$q_1$	$q_4$	$q_5$	
bit 1 is 1:		$q_2$	$q_3$	$q_6$	$q_7$
bit 2 is 0:	$q_0$	$q_1$	$q_2$	$q_3$	
bit 2 is 1:			$q_4$	$q_5$	$q_6$ $q_7$

Now  $\text{cb}(X)$  is  $\mathcal{Q}$  because each element of  $\mathcal{Q}$  is the gcd of 3 well-chosen elements of  $X$ .

## Computing a coprime base

Given a finite set  $S \subset \mathbf{Z}_{>0}$ ,  
compute  $\text{cb}(S)$ :

If  $\#S = 1$ , output  $S$ .

Otherwise, split  $S$  into two halves,  
recursively compute coprime bases,  
and merge those.

## Factoring over a coprime base

Let  $\mathcal{P}$  be a coprime set and  $x \in \langle \mathcal{P} \rangle$ .

Factoring  $x$  over  $\mathcal{P}$ :

If  $\#\mathcal{P} = 1$  and  $p \in \mathcal{P}$ , determine  $k$  such that  $p^k = x$ .

Otherwise, split  $\mathcal{P}$  in two halves.

Compute the factor of  $x$  (not) to use for each half with the “ppo” subalgorithm, and recursively compute the factorization of  $x$ .

# Power detection

Let  $x \in \mathbf{Z}_{>0}$ .

Question: what is the largest integer  $k$  for which  $x$  is a perfect  $k$ -th power?

Can be answered efficiently using coprime bases.

(D.J. Bernstein, H.W. Lenstra, J. Pila)

- For each prime power  $q$  with  $2^q \neq x$ , find an integer  $r_q$  such that *if*  $x$  is a  $q$ -th power, then  $r_q^q = x$ .
- Find a coprime base for  $\{x, r_2, r_3, r_4, \dots\}$ .
- Factor  $x$  over this coprime base.
- Since all roots of  $n$  can also be factored over the same base,  $k$  is the gcd of the exponents in the factorization of  $x$ .

## Applying coprime bases

Recall, input of algorithm:

$n \in \mathbf{Z}_{>1}$ ,  $S \subset \mathbf{Q}^*$ ,  $S$  finite,  $S \neq \emptyset$ .

Compute a coprime base  $\mathcal{P}$  of  
denominators in  $S \cup$  numerators in  $S \cup \{n, 2\}$ .

Detect and reduce perfect powers in  $\mathcal{P}$ .

Factor  $n$  and elements of  $S$  over  $\mathcal{P} \cup \{-1\}$ .

# Counting automorphisms

Recall:

$$B = \langle \mathbf{Q}^*, \zeta_{nw}, \sqrt[n]{S} \rangle$$

Want to compute  $\#\text{Aut}_{\langle \mathbf{Q}^*, \zeta_{nw} \rangle}(B)$ .

Since  $\zeta_{nw} \in B$ , we can ignore the sign of elements in  $S$  here. Assume all elements of  $S$  are positive (for now).

**Theorem.**

$$\text{Aut}_{\langle \mathbf{Q}^*, \zeta_{nw} \rangle}(B) \cong \text{Hom}(B/\langle \mathbf{Q}^*, \zeta_{nw} \rangle, \mu_n).$$

Since  $B/\langle \mathbf{Q}^* \rangle$  has exponent  $n$ , the size of this homomorphism-group is equal to  $[B : \langle \mathbf{Q}^*, \zeta_{nw} \rangle]$ .

## Computing the index

$$B = \langle \mathbf{Q}^*, \zeta_{nw}, \sqrt[n]{S} \rangle$$

$$\#\text{Aut}_{\langle \mathbf{Q}^*, \zeta_{nw} \rangle}(B) = [B : \langle \mathbf{Q}^*, \zeta_{nw} \rangle].$$

By taking  $n$ -th powers, we see this equals  $[\langle S, \mathbf{Q}^{*n}, -1 \rangle : \langle \mathbf{Q}^{*n}, -1 \rangle]$ .

We can further restrict to  $\mathcal{P}$ , and get  $[\langle S, \mathcal{P}^n \rangle : \langle \mathcal{P}^n \rangle]$ .

Note that  $\mathcal{P}$  contains no perfect powers.

Since  $S$  is factored over  $\mathcal{P}$ , this can be computed with linear algebra over  $\mathbf{Z}$ .

# Entanglement

Recall:

$$B = \langle \mathbf{Q}^*, \zeta_{nw}, \sqrt[n]{S} \rangle$$

Define:

$$B_{\text{ab}} = \{x \in B : x^2 \in \zeta_n \mathbf{Q}^*\}.$$

**Theorem.**  $E(B) = E(B_{\text{ab}})$ .

Define:

$$W = \{x \in B : x^2 \in \mathbf{Q}^*_{>0}\} \subset B_{\text{ab}}.$$

**Theorem.**

$$E(B_{\text{ab}}) \cong \text{Gal}(\mathbf{Q}(\zeta_{nw}) \cap \mathbf{Q}(W) / \mathbf{Q}).$$

## Computing entanglement

$$E(B_{ab}) \cong \text{Gal}(\mathbf{Q}(\zeta_{nw}) \cap \mathbf{Q}(W)/\mathbf{Q}).$$

Computing this degree:  
linear algebra over  $\mathbf{F}_2$ :

The set of square roots in  $\mathbf{Q}(W)$  is a sub- $\mathbf{F}_2$ -module of  $\langle \mathcal{P} \rangle / \langle \mathcal{P} \rangle^2$ , as is the set of square roots in  $\mathbf{Q}(\zeta_{nw})$ .

The intersection of these two  $\mathbf{F}_2$ -vector spaces can be efficiently computed.

## Roots of unity

Recall:

$$B' = \langle \mathbf{Q}^*, \sqrt[n]{S} \rangle$$

$$B = \langle B', \zeta_{nw} \rangle$$

We have now computed  $[\mathbf{Q}(B) : \mathbf{Q}(\zeta_n)]$ .

How to get back to  $\mathbf{Q}(B')$  ?

Assume that  $\zeta_{nw} \notin B'$ .

$\text{Aut}_{B'}(B)$  has one non-trivial automorphism,  $\sigma$ .

We want to know if  $\sigma \in \text{Gal}(\mathbf{Q}(B)/\mathbf{Q})$ .

Strategy: map  $\text{Aut}_{B'}(B)$  to  $E(B)$  and check if it is trivial.

## Extra roots of unity

Recall:  $E(B) \cong \text{Gal}(\mathbf{Q}(\zeta_{nw}) \cap \mathbf{Q}(W)/\mathbf{Q})$ .

The map from  $\text{Aut}_{\mathbf{Q}^*}(B)$  to  $E(B)$  is the difference of two natural maps:

$$\begin{aligned} \text{Aut}_{\mathbf{Q}^*}(B) &\rightarrow \text{Aut}(\mu_{nw}) \rightarrow E(B) \\ \text{Aut}_{\mathbf{Q}^*}(B) &\rightarrow \text{Aut}_{\mathbf{Q}^*}(W) \rightarrow E(B) \end{aligned}$$

The image of  $\text{Aut}_{B'}(B)$  in  $E(B)$  by the top map has invariants  $\mathbf{Q}(W \cap \mathbf{Q}(\zeta_n))$ .

The image by the bottom map has invariants  $\mathbf{Q}(W \cap \mathbf{Q}(\zeta_{nw}) \cap B')$ .

Computing  $W \cap B'$  must be done in  $\langle \mathcal{P}, -1 \rangle / \langle \mathcal{P} \rangle^2$ .

All intersections can then be computed using linear algebra over  $\mathbf{F}_2$ .

## Conclusion

Recall:

**Theorem.**

$\text{Gal}(\mathbf{Q}(B)/\mathbf{Q})$  is a normal subgroup of  $\text{Aut}_{\mathbf{Q}^*}(B)$  with abelian cokernel  $E(B)$ .

We computed  $\#E(B)$ ,  $\#\text{Aut}_{\mathbf{Q}^*}(B)$  up to  $\varphi(n)$ , and  $[\mathbf{Q}(B) : \mathbf{Q}(\sqrt[n]{S})]$ .

Combining these gives us the required field degree  $[\mathbf{Q}(\sqrt[n]{S}) : \mathbf{Q}(\zeta_n)]$ .