On the computation of norm residue symbols

Bouw, J.

Chapter 1

Introduction

Let p be a prime number, denote by $\mathbf{Q}_p$ the field of p-adic numbers, and by $\bar{\mathbf{Q}}_p$ an algebraic closure of $\mathbf{Q}_p$. Let F be a finite extension of $\mathbf{Q}_p$ inside $\bar{\mathbf{Q}}_p$ and let F^{ab} be the maximal abelian extension of F inside $\bar{\mathbf{Q}}_p$. Local class field theory gives us a group homomorphism $\phi_F : F^* \longrightarrow \mathrm{Gal}(F^{\mathrm{ab}}/F)$, the *reciprocity map*. For an extensive treatment of the reciprocity map and the broader context of local class field theory, we refer to [**2**], part 2 or [**18**], Teil 2.

Let m be a positive integer and let F contain the m-th roots of unity, which are the elements of $\mu_m = \{x \in \bar{\mathbf{Q}}_p : x^m = 1\}$. The m-th *norm residue symbol* is the map $(\cdot, \cdot)_m : F^* \times F^* \longrightarrow \mu_m$ defined on every pair of elements $\alpha, \beta \in F^*$ by

$$(\alpha, \beta)_m = \frac{\phi_F(\alpha)(\sqrt[m]{\beta})}{\sqrt[m]{\beta}}.$$

The main purpose of this thesis is to prove the following theorems.

THEOREM 1.1. *There is a polynomial-time algorithm that, given a prime number p, a positive integer m and a finite extension F of $\mathbf{Q}_p$ containing a primitive m-th root of unity and also given two elements $\alpha, \beta \in F^*$, computes the norm residue symbol $(\alpha, \beta)_m$.*

At the end of the present introduction we shall describe how the field F and its elements α and β are supposed to be "given" to the algorithm, and how the output is represented. All this will necessarily be done in finite precision, and, as discussed below, this precision should be large enough to guarantee that the output of the algorithm is well-defined. The same comments apply to Theorems 1.2 and 1.4 below. The proof of Theorem 1.1 is found in Section 5 of Chapter 5.

Algorithms for computing norm residue symbols are useful in several contexts. In local class field theory, the norm residue symbol detects which elements are norms from certain extensions (see Remark 5.2). In algebraic number theory, they can be used in the computation of higher power residue symbols in algebraic number fields, see [**4**]. Norm residue symbols are also encountered in arithmetic geometry. For example, the quadratic norm residue symbol $(\alpha, \beta)_2$, which is known as the *Hilbert symbol*, is equal to 1 if and only if the conic $\alpha x^2 + \beta y^2 = z^2$ has an F-rational point. For general m, the norm residue symbol can be used to compute elements in Brauer groups, as explained in [**15**, Section 15]. This can be helpful in detecting the presence of so-called Brauer-Manin obstructions in arithmetic geometry (see [**20**, Chapter 8, Section 2]).

It is hard to find a computer algebra system that allows the possibility of computing norm residue symbols, especially in the case that $m > 2$. In some systems one can approach the problem in an indirect manner, which does not in all cases work out efficiently. We expect that the algorithm that underlies Theorem 1.1 is perfectly suitable for actual implementation.

THEOREM 1.2. *There is a polynomial-time algorithm that, given a prime number p, a positive integer n, and a finite extension F of $\mathbf{Q}_p$, decides whether F contains a primitive p^n-th root of unity and if so, computes such a root of unity.*

The proof of Theorem 1.2 can be found in the last section of Chapter 4. We remark that if $n = 1$, the decision whether F contains a primitive p-th root of unity is a simple verification (see Algorithm 4.13), but if $n > 1$ we perform extensive computations (see Algorithms 4.23 and 4.24) in order to decide whether the required root of unity exists and if so compute it. It is an interesting question whether there exists a faster algorithm than ours in the case that $n > 1$.

The computation of an m-th norm residue symbol can be reduced to two special cases, the *tame* one in which the prime number p does not divide m and the *wild* case in which m is a power of p. In the tame case (see Section 3 of Chapter 5), there is a formula usable in practice to compute the norm residue symbol and also good enough to prove Theorem 1.1. In this thesis we will mainly consider the wild case (see Section 4 of Chapter 5). In that case there are also formulas that can be used to compute the norm residue symbol (see [**7**]), but it remains a challenge to decide whether these formulas can be evaluated in polynomial time and to compare the efficiency of such a computation with the efficiency of our algorithm.

Let p be a prime number, let n be a positive integer and let the field F be a finite extension of $\mathbf{Q}_p$ containing μ_{p^n}. We denote by $\mathrm{ord}_F : F \longrightarrow \mathbb{Z} \cup \{\infty\}$ the surjective valuation function on F. A *prime element* π of F is defined by the property $\mathrm{ord}_F(\pi) = 1$. In the appendix of Milnor's "Introduction to Algebraic K-theory", see [**15**], a *distinguished unit* δ in F is defined by the following properties:

i. $\mathrm{ord}_F(\delta - 1) = \frac{p \cdot \mathrm{ord}_F(p)}{p-1}$,
ii. $\delta \notin (F^*)^p$.

Such a distinguished unit δ has the property that for every unit u of the ring of integers $\mathcal{O}_F$ of F, the norm residue symbol $(u, \delta)_{p^n}$ is a p-th power in the group of p^n-th roots of unity, so $(u, \delta)_{p^n}^{p^{n-1}} = 1$, without δ itself being a p-th power.

The algorithm underlying Theorem 1.1 in the wild case is motivated by a theorem of Moore (see [**15**], Appendix, Theorem A.14). This theorem implies that for any prime element π of F and any distinguished unit δ the symbol $(\pi, \delta)_{p^n}$ generates the cyclic group μ_{p^n}. It also implies that for every pair of elements $\alpha, \beta \in F^*$ the integer $i \in \mathbb{Z}/p^n\mathbb{Z}$ for which $(\alpha, \beta)_{p^n} = (\pi, \delta)_{p^n}^i$ can be computed if $F, p, n, \alpha, \beta, \pi$ and δ are given. Only a few arithmetic rules, which hold for all elements in F^*, are used in the computation. These rules are the following:

i. $(\alpha, \beta)_{p^n} = 1$ if $\alpha + \beta = 1$,
ii. $(\alpha, \beta)_{p^n}^{p^n} = 1$,
iii. $(\alpha_1 \cdot \alpha_2, \beta)_{p^n} = (\alpha_1, \beta)_{p^n} \cdot (\alpha_2, \beta)_{p^n}$,
iv. $(\alpha, \beta_1 \cdot \beta_2)_{p^n} = (\alpha, \beta_1)_{p^n} \cdot (\alpha, \beta_2)_{p^n}$.

In his article “On Computations in Kummer Extensions” (see [**6**]) Daberkow was the first to use these ideas. The proof of Moore's theorem, as given in [**15**], offered him an algorithm to compute the integer i. With this result there are two problems left in the computation of the norm residue symbol.

The first problem is the polynomiality of the algorithm, which is not a part of the discussion in Daberkow's article. Our own algorithm for computing i, while still inspired by [**15**], is very different from Daberkow's, and it does run in polynomial time. It makes use of a presentation for the group $U_1 = \{u \in F : \mathrm{ord}_F(u-1) > 0\} = 1 + \mathfrak{m}$ of *principal units* of F, where $\mathfrak{m} = \pi\mathcal{O}_F$ is the maximal ideal of $\mathcal{O}_F$. The algorithm that proves Theorem 1.2 depends on the same presentation.

The second problem is that knowing the value of i is not the same as knowing the norm residue symbol $(\alpha,\beta)_{p^n} = (\pi,\delta)_{p^n}^i$ as long as we do not know the value of $(\pi,\delta)_{p^n}$. Daberkow does not address this issue. In Chapter 5 of this thesis we compute the true value of the norm residue symbol by using a functorial property of the reciprocity map.

In Chapter 6 we prove the existence of a distinguished unit ϵ with the additional property that $(u,\epsilon)_{p^n} = 1$ if u a unit, which for $n > 1$ is not necessarily the case with a distinguished unit as defined above. Such a distinguished unit will be called a *strongly distinguished unit.*

One can show that a distinguished unit ϵ is strongly distinguished if and only if the field extension $F(\sqrt[p^n]{\epsilon})$ of F, which has degree p^n, is unramified (see Lemma 6.2). In addition, among all elements $\alpha \in F$ for which $F(\sqrt[p^n]{\alpha})$ is unramified of degree p^n over F, the strongly distinguished units are exactly those that are as close as possible to 1. This is a consequence of the following theorem, which also implies that strongly distinguished units exist. It is proved in Chapter 6.

Theorem 1.3. *Let p be a prime number and n a positive integer. Let F be a finite extension of the field $\mathbf{Q}_p$ containing ζ_{p^n}, a primitive p^n-th root of unity. Then there exists $\epsilon \in F$ such that*

i. $\mathrm{ord}_F(\epsilon - 1) = \frac{p}{p-1} \cdot \mathrm{ord}_F(p)$,
ii. *$F(\sqrt[p^n]{\epsilon})$ is an unramified field extension of F of degree p^n.*

There does not exist $\epsilon \in F$ satisfying (ii) *and* $\mathrm{ord}_F(\epsilon - 1) > \frac{p}{p-1} \cdot \mathrm{ord}_F(p)$.

A second result, which is also proved in Chapter 6, tells us that a strongly distinguished unit can be computed in polynomial time.

Theorem 1.4. *There is a polynomial-time algorithm that, given a prime number p, a positive integer n, and a finite extension F of $\mathbf{Q}_p$ containing the p^n-th roots of unity, computes an element ϵ of F satisfying conditions* (i) *and* (ii) *from Theorem 1.3.*

Once a strongly distinguished unit ϵ is available, one may simplify the algorithm underlying Theorem 1.1 by using a formula (see Chapter 6, Lemma 6.3ii) that depends on the property that $(u,\epsilon)_{p^n} = 1$ for every unit u. Thus, if one needs to compute a large number of norm residue symbols in the same field F, it may be of advantage to start by computing a strongly distinguished unit once and for all, using Theorem 1.4.

Moreover, the norm residue symbol $(\pi, \epsilon)_{p^n}$ can also be computed once and for all, and its value is independent of the choice of the prime element π (see Lemma 6.3i).

As announced earlier we will now explain how our field F is given to the algorithms of Theorem 1.1, 1.2 and 1.4, and how we are able to specify the input α, β to the algorithm of Theorem 1.1 using only a finite number of bits. Likewise we will specify in which manner and to which precision the roots of unity and the strongly distinguished units computed by our algorithms are represented.

Let F be any finite extension of $\mathbf{Q}_p$, with no assumptions on roots of unity. We summarize some facts from the standard theory of local fields (see [**24**], Chapter 3). Let f be the degree of the residue class field $\mathcal{O}_F/\mathfrak{m}$ over the prime field $\mathbf{F}_p$ and let $\mathbf{Z}_p$ denote the ring of p-adic integers. There is a monic polynomial $g \in \mathbf{Z}_p[X]$ of degree f that is irreducible modulo p,with the following property: adjoining a root γ of g to $\mathbf{Q}_p$ gives the maximal unramified subfield $E = \mathbf{Q}_p(\gamma)$ of F and $\mathcal{O}_E = \mathbf{Z}_p[\gamma]$ is its ring of integers. There is also a polynomial $h \in \mathbf{Z}_p[X, Y]$ such that $h(\gamma, Y) \in E[Y]$ is a monic and irreducible polynomial of degree $e = \operatorname{ord}_F(p)$ with the following properties: first, it satisfies specific conditions on its coefficients (see Chapter 3, Section 3) that make it into an *Eisenstein polynomial*; and second, it has a zero π in F. Then it is automatic that $F = E(\pi)$, that F is totally ramified over E with prime element π, and that $\mathcal{O}_F = \mathbf{Z}_p[\gamma, \pi] \cong \mathbf{Z}_p[X, Y]/(g, h)$.

Because F is the field of fractions of $\mathcal{O}_F$, it suffices to "give" $\mathcal{O}_F$ instead of F. However, in algorithms we cannot work with elements of $\mathcal{O}_F$ in infinite precision, so we use an approximation of $\mathcal{O}_F$, good enough for our purposes. This approximation is the finite ring $\mathcal{O}_N = \mathcal{O}_F/\mathfrak{m}^N$, where $N \in \mathbf{Z}_{>0}$ is the precision, to be chosen large enough as discussed below. If the polynomials g_N and h_N satisfy $g_N \equiv g \pmod{p^{\lceil \frac{N}{e} \rceil}}$ and $h_N \equiv h \pmod{p^{\lceil \frac{N}{e} \rceil}}$ then we have $\mathcal{O}_N \cong (\mathbf{Z}/p^{\lceil \frac{N}{e} \rceil}\mathbf{Z})[X, Y]/(g_N, h_N, Y^N)$, with γ and π corresponding to X and Y respectively (see Chapter 3, Section 4.1). Then our field is "given" in precision N by p, g_N and h_N.

Any element $x \in \mathcal{O}_N$ is represented by a sum of the form $\sum_{i=0}^{N-1} c_i \pi_i$, where π_i is a certain element with $\operatorname{ord}_F(\pi_i) = i$ (see Definition 2.3), and where each c_i belongs to the set $\mathcal{C} = \{\sum_{j=0}^{f-1} d_j \gamma^j : d_j \in \{0, 1, \ldots, p-1\} \text{ for each } j\}$ of *digits* (see Definition 2.2). Observe that each coset of $\mathcal{O}_F/\mathfrak{m}$ contains exactly one digit. The elements of $(\mathcal{O}_N)^*$ are characterised by the property that $c_0 \neq 0$. This representation of elements of $(\mathcal{O}_N)^*$ will be used below, and it also applies to the roots of unity and strongly distinguished units that are computed by our algorithms. Note that $O(N \log q)$ bits suffice to represent any element of $\mathcal{O}_N$, where $q = p^f = \#\mathcal{C}$ is the number of elements of the residue field $\mathcal{O}_F/\mathfrak{m}$. Every arithmetical operation performed in our algorithms takes place in $\mathcal{O}_N$ for some N or in the ring $\mathbf{Z}$.

We will specify α and β in Theorem 1.1 using the analogue for F^* of *scientific notation.* This will do justice to the multiplicative nature of the norm residue symbol and also accommodate elements that do not belong to $\mathcal{O}_F^*$. Just as every positive real number can be uniquely written as $u \cdot 10^a$ with $u \in [1, 10)$ and $a \in \mathbf{Z}$, so can each element of F^* be uniquely written as $u \cdot \pi^a$ with $u \in (\mathcal{O}_F)^*$ and $a \in \mathbf{Z}$. We need to turn this notation into one that uses only a finite number of bits.

As in Theorem 1.1, let $m \in \mathbf{Z}_{>0}$ be such that $\mu_m \subset F$. Since the value of $(\alpha, \beta)_m$ depends only on the cosets $\alpha(F^*)^m$, $\beta(F^*)^m \in F^*/(F^*)^m$ (see Chapter 5, Proposition

5.1), it will for our purposes suffice to represent elements of $F^*/(F^*)^m$, and this is what can be done with a finite number of bits, as follows. If $u \cdot \pi^a \in F^*$ is as above, then knowing the coset $u \cdot \pi^a \cdot (F^*)^m$ is clearly equivalent to knowing a modulo $m\mathbf{Z}$ and u modulo $(\mathcal{O}_F^*)^m$. Now assume that our precision satisfies $N \geq 1$ in the tame case (see Algorithm 5.4) and $N \geq \frac{e}{p-1} + \mathrm{ord}_F(m) + 1$ otherwise. Then the group $1 + \mathfrak{m}^N$ is contained in $(\mathcal{O}_F^*)^m$ (see Chapter 4, Corollary 4.9), so we have a surjective group homomorphism

$$(\mathcal{O}_N)^* = \mathcal{O}_F^*/(1 + \mathfrak{m}^N) \to \mathcal{O}_F^*/(\mathcal{O}_F^*)^m.$$

Hence we can represent elements of $F^*/(F^*)^m$ by pairs $(\bar{a}, \bar{u}) \in \mathbf{Z}/m\mathbf{Z} \times (\mathcal{O}_N)^*$ with $(\bar{a}, \bar{u})$ representing the coset $u \cdot \pi^a (F^*)^m$, and that is what we shall do (see Chapter 5, section 2). The total number of bits used is $O(N \log q + \log m)$.

In Theorem 1.2 we choose the precision N in which our field F is given such that the inequality $N \geq \frac{e}{p-1} + e \cdot n + 1$ is satisfied. The precision of the output is $N - e \cdot n$ (see Algorithm 4.24, Proposition 4.25 and Theorem 4.26). We remark that due to the fact that in our algorithm p-th roots of principal units are computed, the precision of the output will be smaller than the precision of the input. In fact, the precision of the output is just large enough to distinguish between different p^n-th roots of unity and therefore the root of unity computed by the algorithm is well-defined. In Theorem 1.4 the precision of the input is also required to satisfy $N \geq \frac{e}{p-1} + e \cdot n + 1$, and the precision of the output is N itself (see Algorithm 6.8 and Proposition 6.9). In Theorem 1.1 we have to distinguish two cases. In the tame case, we require $N \geq 1$ for the precision of the input, and the precision of the output equals N (see Algorithm 5.4 and Proposition 5.5). In the other case, we choose the precision N of the input such that $N \geq 3(r+1)e + 1$, where r is the integer for which $p^r \mid\mid e$ and the precision of the output is $N - (r+1)e$ (see Algorithm 5.24, Proposition 5.25 and Theorem 5.26).