COMPUTING LOCAL ARTIN SYMBOLS IN MAGMA

WILLIAM B. HART AND SAMIR SIKSEK

Abstract. We give MAGMA code for computing local Artin symbols in a relative extension of number fields.

1. The Algorithm

We wish to compute local Artin symbols in an extension L/K of number fields. We implement this algorithm in MAGMA using the algorithm presented in [1].

1.1. **Input to the algorithm.** As input to the algorithm we supply:

- An abelian extension L/K of number fields
- A valuation v corresponding to a prime ideal $\mathfrak{p}$ of K
- An element a_v of K_v^* for which we wish to compute the Artin symbol $(a_v, L_w/K_v)$

To specify the abelian extension L/K, we supply K by a minimum polynomial of a generator of K over $\mathbb{Q}$ and L by a polynomial defining L as a relative extension of K.

The valuation v will be given by specifying a prime p of K.

In this document we will supply a_v by specifying an element a of K^* (which is a special case). More generally, one may specify an element a_v by giving the set of coefficients of its expansion

$$a_v = \sum_{i=\mathrm{ord}_v(a_v)} c_i \pi_v^i, \tag{1}$$

truncated at an appropriate precision (discussed below), where π_v is a uniformizing element of v (thought of as belonging to K^*) and the coefficients c_i are representatives in K of appropriate elements of the residue field $K_v^*/(\pi_v)$.

In fact the algorithm presented below, if given an element a of K^* will first generate an expansion of the form (1), so that computing an Artin symbol from an expansion of the latter kind is really a subset of what is presented below.

Let us give an example of how to specify these different components to MAGMA.

We begin by specifying a base field K. We need to give a minimum polynomial for a generator of K over $\mathbb{Q}$, and so first we set up a polynomial ring over the integers $\mathbb{Z}$. Here y will represent the generator of the number field K over $\mathbb{Q}$. It is also useful to specify O to be the ring of integers of K.

```
R<x> := PolynomialRing(Integers());
f := x^2-10;
K<y> := NumberField(f);
O := MaximalOrder(K);
```

Next we specify the extension L of K by giving a minimal polynomial for a generator of L/K. As we wish to work with the functions in MAGMA which apply to abelian extensions, we tell MAGMA to consider L to be an abelian extension of K. This of course assumes that this is the case.

```
L:=ext<K|x^3+x^2-2*x-1>;
L:=AbelianExtension(L);
```

Next we wish to specify a prime p of K which will correspond to our valuation v. Any method for specifying a prime ideal of K in MAGMA will work here. In this example we simply take p to be one of the primes of K that lie above 13.

```
I := Decomposition(O,13);
p := I[1,1];
```

Finally we specify our element $a \in K^*$ for which we wish to compute the Artin symbol $(a_v, L_w/K_v)$. Again any way of specifying an element of K^* in MAGMA will do. Here we chose the element $a = 169\sqrt{10} + 221$ of K.

```
a := elt<O|169,221>;
```

1.2. **The Implementation of the Algorithm.** First of all, we need to compute an admissible cycle $\mathfrak{c}$ for L/K. The conductor of the extension will do.

```
c,pinf:=Conductor(L);
```

We compute $\mathrm{ord}_v(a)$ and $\mathrm{ord}_v(\mathfrak{c})$. The precision s to which we need to compute the expansion (1) is the sum of these two orders.

```
ordva:=Valuation(a,p);
ordvc:=Valuation(c,p);
s := ordva+ordvc;
```

Now we are able to compute the $\mathfrak{p}$-adic expansion of a. To compute the expansion, we first compute a uniformizing element μ for the ideal $\mathfrak{p}$.

The coefficients c_i will be stored in the sequence ci, e.g. $ci[1]$ will be the coefficient c_0 of the expansion (1).

```
mu := UniformizingElement(p);

ci:=[0:x in [0..s]];
rem:=a;
ci[1]:= a mod p;
for i := 1 to s do
rem:=rem-ci[i]*mu^(i-1);
ci[i+1] := O!(rem/mu^i) mod p;
end for;
```

We are now able to compute the Artin symbol we are after if $\mathfrak{p} \nmid \mathfrak{c}$. It is returned in **symbol**.

```
mA := ArtinMap(L);

if ordvc eq 0 then
symbol:=mA(p)^ordva;
symbol;
end if;
```

If this is not the case, the algorithm continues. We truncate the expansion for a one term early, calling the result $h1$. Then we find an $h \in K$ such that $h \equiv h1 \pmod{p^s}$ and $h \equiv \mod \mathfrak{c}$ using the Chinese Remainder Theorem.

```
if ordvc ne 0 then
h1 := 0;
for i := 0 to s-1 do
h1 := h1+ci[i+1]*mu^i;
end for;

h:= CRT([O!h1,O!1],[p^s,c]);
```

Now we need h to be adjusted so that $h \equiv \mod^* \mathfrak{c}$, which means we have to ensure that the archimedean part of the mod* condition holds, i.e. that the value h is positive at each of the real embeddings **pinf** that we obtained when we computed the conductor.

In order to achieve this, we find a rational integer which is zero modulo $\mathfrak{p}^s\mathfrak{c}$ (the norm of this ideal will do) and add it to h until h satisfies the required mod* property.

```
if #pinf ne 0 then
infp:=InfinitePlaces(K);
alpha := Norm(p^s*c);
i:=0;
repeat
h:=h+(K!(2^i))*alpha;
positive:=true;
for j:=1 to #pinf do
if Evaluate(h,infp[j]) le 0.01 then
positive:=false;
end if;
end for;
i:=i+1;
until positive eq true;
end if;
```

Finally we compute the required Artin symbol.

```
aideal := (h*O)*p^(-Valuation(h,p));
symbol := mA(aideal)^(-1);
end if;
```

References

[1] Acciaro, Vincenzo and Klüners, Jürgen *Computing Local Artin Maps, and Solvability of Norm Equations* J. Symb. Comp. (2000) **11**, 1-14

E-mail address: w.b.hart@maths.warwick.ac.uk