Efficient Computation of Minimal Polynomials in Algebraic Extensions of Finite Fields*

Victor Shoup
IBM Zurich Research Lab, Säumerstr. 4, 8803 Rüschlikon, Switzerland
`sho@zurich.ibm.com`

March 11, 1999

Abstract

New algorithms are presented for computing the minimal polynomial over a finite field K of a given element in an algebraic extension of K of the form $K[\alpha]$ or $K[\alpha][\beta]$. The new algorithms are explicit and can be implemented rather easily in terms of polynomial multiplication, and are much more efficient than other algorithms in the literature.

1 Introduction

In this paper, we consider the problem of computing the minimal polynomial over a finite field K of a given element σ in an algebraic extension of K of the form $K[\alpha]$ or $K[\alpha][\beta]$. The minimal polynomial of σ is defined to be the unique monic polynomial $\phi_{\sigma/K} \in K[x]$ of least degree such that $\phi_{\sigma/K}(\sigma) = 0$.

In the first case, we assume that the ring $K[\alpha]$ is given as $K[x]/(f)$ where $f \in K[x]$ is a monic polynomial of degree n, and that elements in $K[\alpha]$ are represented in the natural way as elements of $K[x]_{<n}$ (the set of polynomials of degree less than n).

Similarly, in the second case, we assume that $K[\alpha]$ is given as above, and that $K[\alpha][\beta]$ is given as $(K[\alpha][y])/(F)$, where $F \in K[\alpha][y]$ is a monic polynomial of degree m.

*This is a corrected revision of the November 3, 1998 version, and will appear in ISSAC '99

We give new algorithms for both cases. To simplify the presentation, we let $R = K[\alpha]$ in the first case, and $R = K[\alpha][\beta]$ in the second case, and we let $d = \dim_K R$ in either case.

In both cases, our new algorithms have a running time of $O(M(d)d^{1/2} + d^2)$ arithmetic operations in K. Here, $M(d)$ is a bound on the number of arithmetic operations in K used by an algorithm to multiply elements in $K[x]_{<d}$. In stating this running-time bound, we make the (justifiable) assumption that multiplication in R takes $O(M(d))$ operations in K.

The space requirement of our algorithms is $O(d^{3/2})$ elements of K.

We stress that these algorithms are quite practical, and easily beat all competing algorithms in terms of running times for inputs of modest size. Moreover, they can be easily implemented using existing algorithms for polynomial multiplication. Indeed, we have implemented these algorithms, and have verified hat our running-time analysis is an accurate predictor of the actual behavior. Also, our algorithms *do not* require that $K[\alpha]$ or $K[\alpha][\beta]$ are fields.

1.1 Applications

Of course, computation of minimal polynomials is a fundamental operation, interesting in its own right.

Additionally, computing minimal polynomials in $K[\alpha]$ is an important step in efficient algorithms for factoring univariate polynomials over K (see [9] for details). This particular example illustrates the need for considering the case where $R = K[\alpha]$ is not a field, since $K[\alpha] = K[x]/(f)$, where f is the polynomial we are trying to factor.

Also, an algorithm for computing minimal polynomials in $K[\alpha]$ or $K[\alpha][\beta]$ can be used as a subroutine in a modular algorithm for computing minimal polynomials over $\mathbf{Z}$ of integral extensions of $\mathbf{Z}$ of an analogous form: $\mathbf{Z}[u]$ and $\mathbf{Z}[u][v]$. This example also illustrates the need for treating non-fields R, since even if we start with an integral domain $\mathbf{Z}[u]$ or $\mathbf{Z}[u][v]$, after reduction modulo a prime, we may not end up with a field; moreover, just testing for that condition is more expensive than our algorithms for computing minimal polynomials.

1.2 Relation to Other Algorithms

Of course, the simplest algorithm to use is Gaussian elimination: given $\sigma \in R$, compute the sequence

$$1, \sigma, \sigma^2, \ldots, \sigma^d,$$

and find a linear relation over K using Gaussian elimination. Using standard algorithms, this results in a running time of $O(d^3)$ operations in K, and space for $O(d^2)$ elements in K. We do not consider asymptotically fast matrix methods here, as these are of no practical significance. Clearly, our new algorithms are substantially less costly in terms of both time and space.

Special algorithms can be used when and $R = K[\alpha]$ is also a field. Suppose K has cardinality q. Then an algorithm of Gordon [3] runs as follows. Given $\sigma \in R$, we compute the sequence

$$\sigma, \sigma^q, \ldots, \sigma^{q^{l-1}},$$

where l is the smallest positive integer such that $\sigma^{q^l} = \sigma$, so $l \mid d$, and $l = d$ in the worst case. The algorithm then computes

$$(\alpha - \sigma)(\alpha - \sigma^q) \cdots (\alpha - \sigma^{q^{l-1}}) \in R,$$

from which $\phi_{\sigma/K}$ is easily obtained.

Using standard exponentiation algorithms, the running time of this algorithm is

$$O(\log q M(d) d)$$

operations in K, and the space requirement is $O(d)$ elements in K.

Although this algorithm has a lower space requirement than ours, its running time is significantly higher than ours. Moreover, this algorithm does not work in the case where R is not a field, nor does it work in the case where $R = K[\alpha][\beta]$.

Our new algorithm is based on Wiedemann's projection method: given $\sigma \in R$, we choose a random projection $\pi \in \hom_K(R, K)$, and compute the sequence

$$c_0 = \pi(1), c_1 = \pi(\sigma), \ldots, c_{2d-1} = \pi(\sigma^{2d-1}).$$

We then feed this projected sequence to an implementation of the Berlekamp/Massey algorithm to compute its minimal polynomial over K (as a linearly generated sequence). In general, this procedure will yield a divisor of $\phi_{\sigma/K}$, and some extra work is required to obtain $\phi_{\sigma/K}$ itself.

A straightforward implementation of this approach takes $O(dM(d))$ operations in K, the bottleneck being the computation of the successive powers of σ. However, we do not really need to explicitly compute these powers—we only need to compute the projections of these powers under the given π. We call this the *power projection problem*.

It was observed in [8] that this problem was related to the *polynomial evaluation problem*: given $g \in K[x]$ of degree less than $2d$ and $\sigma \in R$, compute $g(\sigma) \in R$. More precisely, if the power projection is viewed as a linear operation on the coordinate vector of π, and polynomial evaluation is viewed as a linear operation on the coordinate vector of g, then under the natural choice of basis, the matrices representing these linear transformations are the transpose of one another. An algorithmic theorem, which we call the *transposition principle*, states that any bi-linear algorithm for computing a matrix/vector product can be transformed into one of the same complexity that computes the transposed matrix/vector product. One simply reverses the flow of the linear circuit representing the algorithm (see, e.g., [5]). Furthermore, it was noted that an algorithm of Brent and Kung [2] solved the polynomial evaluation problem using $O(M(d)d^{1/2} + d^2)$ operations in K. All of these observations together implied the existence of an algorithm for the minimal polynomial problem whose running time is $O(M(d)d^{1/2} + d^2)$ operations in K.

So in theory, at least, the problem we set out to solve has already been solved in [8]. However, none of the algorithms in [8] can be directly implemented in any practical way. A straightforward application of that theory would have required the explicit construction of an arithmetic circuit solving the polynomial evaluation problem, followed by a “reverse evaluation” of that circuit to solve the power projection problem. This would hardly be practical for a number of reasons, not the least important of which is that the most practical implementations of polynomial multiplication algorithms often stray outside the model of arithmetic circuits over K.

So the work in [8] left a significant gap between theory and practice. This gap was partially closed in [9] where explicit algorithms for the power projection problem were given in the case that $R = K[\alpha]$ and $K = \mathbf{Z}/(p)$, and also assuming a specific FFT-based implementation of multiplication in $K[x]$.

Our new algorithms for $R = K[\alpha]$ and $R = K[\alpha][\beta]$ are fairly simple, and are described in terms of multiplication in $K[x]$ or $K[\alpha][y]$. This means that they should be quite easy to implement in any computer algebra software environment.

In the case $R = K[\alpha]$, our solution amounts to essentially an explicit algorithm for the power projection problem. Our solution in the case $R = K[\alpha][\beta]$ is a bit different—we do not solve the power projection problem for an explicitly given π, but rather for a random, implicitly defined π, which is good enough.

We remark that Kaltofen [4] also observed that the transposition principle can be brought to bear on the minimal polynomial problem in $K[\alpha][\beta]$.

2 Polynomial Evaluation and Power Projection

In this section, we recall algorithms from the literature for polynomial evaluation and power projection, and extend them to satisfy our particular requirements.

2.1 Polynomial Evaluation

We first consider the polynomial evaluation problem in $K[\alpha]$ over K. That is, given $\sigma \in K[\alpha]$ and $g = \sum_{i=0}^{l-1} g_i x^i \in K[x]$, compute $g(\sigma)$.

This can be done efficiently using an algorithm of Brent and Kung [2] using $O(M(n)l^{1/2} + ln))$ operations in K, as follows. We set $k = \lfloor l^{1/2} \rfloor$ and $k' = \lceil l/k \rceil$. The algorithm first computes the powers $1, \sigma, \ldots, \sigma^k$. Then the following is executed:

$$
\begin{array}{l}
\tau \leftarrow 0 \in K[\alpha] \\
\text{for } i \leftarrow k'-1, \ldots, 0 \text{ do} \\
\quad \tau \leftarrow \tau \cdot \sigma^k + \sum_{j=0}^{k-1} g_{ik+j} \sigma^j \\
\text{output } \tau
\end{array}
$$

It is easy to verify that this algorithm is correct and can be implemented using $O(M(n)l^{1/2} + ln)$ operations in K.

2.1.1 A generalization

We will also need an algorithm for the polynomial evaluation problem in $K[\alpha][\beta]$ over K. That is, given $\sigma \in K[\alpha][\beta]$ and $g \in K[x]_{<l}$, compute $g(\sigma)$. It is straightforward to check that the above algorithm extends to this case, obtaining an algorithm that uses $O(M(d)l^{1/2} + ld)$ operations in K.

2.2 Power Projection

We first consider the power projection problem in $K[\alpha]$ over K: for a given integer l, and a given $\sigma \in K[\alpha]$, and a given vector $v \in K^n$, compute the sequence

$$c_0 = \langle v, 1 \rangle,\ c_1 = \langle v, \sigma \rangle,\ \ldots,\ c_{l-1} = \langle v, \sigma^{l-1} \rangle$$

of l values in K.

Here, $\langle \cdot, \cdot \rangle$ denotes inner product, and where there can be no confusion, we allow one of the arguments to come from a vector space isomorphic to K^n under a natural isomorphism.

This can be done using an algorithm described in [9] using $O(M(n)l^{1/2} + ln)$ operations in K.

To describe that algorithm, for $\tau \in K[\alpha]$ and $v \in K^n$, we define $\tau \circ v \in K^n$ to be the unique vector satisfying $\langle \tau \circ v, \tau' \rangle = \langle v, \tau\tau' \rangle$ for all $\tau' \in K[\alpha]$. This operation makes K^n into a module over $K[\alpha]$.

We call this operation *transposed multiplication in $K[\alpha]$ over K*; indeed, for fixed τ, the matrix representing the K-linear map $v \mapsto \tau \circ v$ is just the transpose of the matrix representing the "multiplication by τ" map. By the transposition principle, then, the complexity of transposed multiplication is the same as that of multiplication. In the next section, however, we give an explicit algorithm for transposed multiplication in terms of polynomial multiplications.

Note that it makes perfect sense to define transposed multiplication over K when K is replaced by an arbitrary commutative ring with unity, and our algorithms work in this case as well.

The algorithm for power projection runs as follows. We set $k = \lfloor l^{1/2} \rfloor$ and $k' = \lceil l/k \rceil$. We first compute $1, \sigma, \ldots, \sigma^k$. Then we execute the following algorithm:

for $i \leftarrow 0, \ldots, k' - 1$ do
  $c_{ik+j} \leftarrow \langle v, \sigma^j \rangle \quad (0 \leq j < k)$
  $v \leftarrow \sigma^k \circ v$

It is readily verified that this procedure will compute the desired sequence. The running time is $O(l^{1/2})$ multiplications in $K[\alpha]$, $O(l^{1/2})$ transposed multiplications in $K[\alpha]$, plus $O(ln)$ operations in K. Thus the total cost is $O(M(n)l^{1/2} + ln)$ operations in K.

2.2.1 A generalization

Besides an algorithm for power projection in $K[\alpha]$ over K, we will also need an analogous algorithm for $K[\alpha][\beta]$ over K. In this case, we formulate the problem as follows. We are given a positive integer l, $\sigma \in K[\alpha]$, $v \in K[\alpha]^m$, and $w \in K^n$. Our goal is to compute the sequence

$$c_0 = \langle w, \langle v, 1 \rangle \rangle,\ \langle w, \langle v, \sigma \rangle \rangle,\ \ldots,\ \langle w, \langle v, \sigma^{l-1} \rangle \rangle$$

of l values in K.

We present here an algorithm that requires $O(M(d)l^{1/2} + ld)$ operations in K. For a vector $v \in K[\alpha]^n$, we let $v[t]$ denote its t-th component, indexing from 0. We also extend this notation to R, expressing elements in R on the natural polynomial basis over $K[\alpha]$.

This algorithm makes use of transposed multiplications in R over $K[\alpha]$, as well as transposed multiplications in $K[\alpha]$ over K.

Let $k = \lfloor l^{1/2} \rfloor$, and $k' = \lceil l/k \rceil$. We first compute $1, \sigma, \ldots, \sigma^k$. Then we execute the following:

```
for i ← 0, ..., k' − 1 do
    for t ← 0, ..., m − 1 do
        w^(t) ← v[t] ∘ w
    for j ← 0, ..., k − 1 do
        c_{ik+j} ← Σ_{t=0}^{m−1} ⟨w^(t), σ^j[t]⟩
    v ← σ^k ∘ v
```

The correctness and running-time bound are easily verified. Note that we cannot simply compute

$$c_{ik+j} \leftarrow \langle w, \langle v, \sigma^j \rangle \rangle \quad (0 \leq j < k)$$

by direct application of these formulas, since then we would get a term of $O(lmM(n))$ in the running time.

3 Transposed Multiplication

To complete the algorithm for power projection described in the previous section, we have to specify an algorithm for transposed multiplication in $K[\alpha]$ over K: given $\tau \in K[\alpha]$ and $v \in K^n$, compute $\tau \circ v \in K^n$.

By the transposition principle, the complexity of this problem is the same as that of the ordinary modular multiplication problem. But we want a specific algorithm.

Let $b = \sum_{i=0}^{n-1} b_i x^i \in K[x]$, where $b(\alpha) = \tau$. Let $v = (v_0, \ldots, v_{n-1})^T$. Recall that $f \in K[x]$ is the monic polynomial of degree n defining the extension $K[\alpha]$ over K. Let $f = x^n + \sum_{i=0}^{n-1} f_i x^i$.

The algorithm is based on the following lemma.

Lemma 1 *We have*

$$\tau \circ v = \begin{pmatrix} v_0 & v_1 & \cdots & v_{n-1} \\ v_1 & v_2 & \cdots & v_n \\ & & \vdots & \\ v_{n-1} & v_n & \cdots & v_{2n-2} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix}, \tag{1}$$

where $v_n, \ldots, v_{2n-2}$ are defined by

$$f_0 v_{i-n} + f_1 v_{i-n+1} + \cdots + f_{n-1} v_{i-1} + v_i = 0 \quad (n \leq i \leq 2n-2). \tag{2}$$

Proof. For $0 \leq i < n$, we have

$$(\tau \circ v)[i] = \langle \tau \circ v, \alpha^i \rangle = \langle \alpha^i \circ v, \tau \rangle.$$

So to prove the lemma, it suffices to show that for $0 \leq i, j < n$,

$$(\alpha^i \circ v)[j] = v_{i+j}.$$

But

$$(\alpha^i \circ v)[j] = \langle \alpha^i \circ v, \alpha^j \rangle = \langle v, \alpha^{i+j} \rangle.$$

So it suffices to show that $\langle v, \alpha^i \rangle = v_i$ for $0 \leq i \leq 2n-2$. For $0 \leq i < n$ this is trivial, and for $i \geq n$, this follows from the fact that the sequence of powers α^i and the sequence of values v_i satisfy the same linear recurrence. □

An algorithm for computing $v_n, \ldots, v_{2n-2}$ as above was presented in [7] (see also [10]). Define

$$a = \sum_{i=0}^{n-1} v_i x^i \quad \text{and} \quad \hat{a} = \sum_{i=0}^{n-2} v_{i+n} x^i.$$

Let $\tilde{f}$ be the "reverse" of f; that is, $\tilde{f} = x^n f(x^{-1})$. Also, let h be the inverse of $\tilde{f}$ modulo x^{n-1}. Then from [7] we can compute $\hat{a}$ as

$$\hat{a} = -h(\tilde{f}a \text{ div } x^n) \bmod x^{n-1}.$$

Once we have $\hat{a}$, we can compute $\tau \circ v$ via (1) as the coefficient vector of the polynomial

$$(\tilde{b}a \text{ div } x^{n-1}) + x \cdot (\tilde{b}\hat{a} \bmod x^{n-1}),$$

where $\tilde{b} = x^{n-1}b(x^{-1})$.

Assuming we have pre-computed h, we can put all this together in the following algorithm:

$$\begin{aligned}
&t_1 \leftarrow \tilde{f} \cdot a \text{ div } x^n \\
&t_2 \leftarrow \tilde{b} \cdot a \text{ div } x^{n-1} \\
&t_3 \leftarrow h \cdot \tilde{b} \cdot t_1 \bmod x^{n-1} \\
&\text{output } t_2 - x \cdot t_3
\end{aligned}$$

In a straightforward implementation, this will require four multiplications in $K[x]_{<n}$. However, if either τ or v is fixed for several computations of $\tau \circ v$, we can pre-condition to reduce the cost. In fact, both of these cases arise in our algorithms for computing minimal polynomials.

3.1 Pre-conditioning on τ

In this case, we can pre-compute $\tilde{b}h \bmod x^{n-1}$, taking one multiplication in $K[x]_{<n}$, and then the above algorithm for transposed multiplication takes just three additional multiplications in $K[x]_{<n}$.

3.2 Pre-conditioning on v

In this case, we can pre-compute t_1 and $ht_1 \bmod x^{n-1}$, taking two multiplications in $K[x]_{<n}$, and then the above algorithm for transposed multiplication takes just two additional multiplications in $K[x]_{<n}$.

3.3 An implementation note

In a given implementation, it may be more convenient to compute $c \cdot h \bmod x^{n-1}$, for a given $c \in K[x]_{<n}$, as $\tilde{c}x^{n-1}$ div f, where $\tilde{c} = x^{n-1}c(x^{-1})$. In a good implementation of polynomial arithmetic, computing $c \cdot h \bmod x^{n-1}$ in this way should cost one multiplication in $K[x]_{<n}$, assuming pre-conditioning on f.

3.4 An open question

We remark that our algorithm for transposed multiplication is not quite as efficient as that implied by the transposition principle. Indeed, an ordinary multiplication in $K[\alpha]$ can be implemented using 3 multiplications in $K[x]_{<n}$ (assuming pre-conditioning on f), whereas our transposed algorithm takes 4 (without pre-conditioning on τ or v). This lack of "optimality" comes from the way we compute the matrix-vector product (1). The matrix here is a Hankel matrix, and we have presented a fairly standard algorithm that computes a Hankel matrix/vector product using two polynomial multiplications in $K[x]_{<n}$. Interestingly, however, the transposition principle implies that one can compute a Hankel matrix/vector product in the same time as *one* multiplication in $K[x]_{<n}$. This suggests the following question: *can a Hankel matrix/vector product calculation be reduced to a single multiplication in $K[x]_{<n}$ via an efficient transformation?*

3.5 A generalization

The above algorithm for transposed multiplication in $K[\alpha]$ over K also works for transposed multiplication in $K[\alpha][\beta]$ over $K[\alpha]$. Indeed, nothing about the above algorithm requires that the base ring is a field, just that the leading coefficient of the polynomial defining the extension is a unit.

4 The case $R = K[\alpha]$

In the algorithms in this section and the next, we shall need as a subroutine the Berlekamp/Massey algorithm [6]. Let $c_0, c_1, \ldots$ be a sequence of elements in K that is linearly generated with minimal polynomial of degree at most l. Then this algorithm takes as input the first $2l$ elements in this sequence, and outputs its minimal polynomial. The running time is $O(l^2)$. One can also use an asymptotically fast version [1] that runs in time $O(M(l)\log l)$, but this is not critical in our application.

We shall denote the output of this algorithm as $\mathrm{BerMass}(c_0,\ldots,c_{2l-1})$.

Now we present and analyze an algorithm for computing the minimal polynomial of an element $\sigma \in K[\alpha]$ over K. The algorithm can be viewed as an instance of a more general algorithm of Wiedemann [11], but it turns out that the transposed multiplication algorithm plays a critical role.

Recall that $n = [K[\alpha] : K]$.

Let $\sigma \in K[\alpha]$ be the polynomial whose minimal polynomial we want to compute. Let l be a bound on the degree of the minimal polynomial of σ. By default, we can always set $l = n$.

The algorithm then runs as follows.

$g \leftarrow 1 \in K[x]$
$\tau \leftarrow 1 \in R$

while $\tau \neq 0$ do
  choose $v \in K^n$ at random
  $v \leftarrow \tau \circ v$
  $c_i \leftarrow \langle v, \sigma^i \rangle$ for $i = 0, \ldots, 2(l - \deg g) - 1$
  $g' \leftarrow \mathrm{BerMass}(c_0, \ldots, c_{2(l-\deg g)-1})$

  $g \leftarrow gg'$
  $\tau \leftarrow \tau g'(\sigma)$

output g

We claim that

(1) upon termination, this algorithm outputs $\phi_{\sigma/K}$;

(2) using the algorithms described above for power projection, polynomial evaluation, and transposed multiplication, along with the Berlekamp/Massey algorithm, the algorithm has an expected running time of
$$O(M(n)l^{1/2} + ln)$$
operations in K.

The first claim follows easily by induction. The induction hypothesis is that at the beginning of each loop iteration, g divides $\phi_{\sigma/K}$, and $\tau = g(\sigma)$. Now, the polynomial g' computed in the loop body is a divisor of the minimal polynomial of the linearly generated sequence

$$\tau, \tau\sigma, \tau\sigma^2, \ldots.$$

Moreover, the minimal polynomial of this sequence is precisely $\phi_{\sigma/K}/g$. From this, it immediately follows that the loop invariant indeed holds from one iteration to the next.

As for the second claim, it is easily verified that a single loop iteration costs at most

$$O(M(n)l^{1/2} + ln)$$

operations in K. Moreover, more general results of Wiedemann [11] imply that the expected number of loop iterations of this algorithm is $O(1)$, from which the second claim is immediate.

The above algorithm can be easily "optimized" in several ways to improve its performance somewhat. For example, we can always quit as soon as $\deg g = l$, perhaps avoiding the computation of $g'(\sigma)$ at the bottom of the loop. Also, in the first iteration of the loop, the instruction $v \leftarrow \tau \circ v$ can be skipped.

Another simplification can be made when it is known in advance that $\phi_{\sigma/K}$ is irreducible; for example, when $K[\alpha]$ is a field. In this case, exactly one iteration of the loop suffices, setting $v = (1, 0, \ldots, 0)^T$.

5 The case $R = K[\alpha][\beta]$

Now we present and analyze an algorithm for computing the minimal polynomial of an element $\sigma \in K[\alpha][\beta]$ over K. We assume $K[\alpha] = K[x]/(f)$, where $f \in K[x]$ is monic, and of degree n, and that $K[\alpha][\beta] = K[\alpha][y]/(F)$, where $F \in K[\alpha][y]$ is monic, and of degree m. Let $d = \dim_K R = nm$.

First, our algorithm needs to construct a vector $w \in K^n$ that satisfies the following property:

> *for all non-zero $\mu \in K[\alpha]$ there exists a $\nu \in K[\alpha]$ such that $\langle w, \mu\nu \rangle \neq 0$.*

For this purpose, we can always take $w = (0, \ldots, 0, 1)^T$. If the constant term of f is nonzero, we can take $w = (1, 0, \ldots, 0)^T$.

Now, for $v \in K[\alpha]^m$, define $\pi_{v,w} : R \to K$ be the K-linear map that sends $\tau \in R$ to $\langle w, \langle v, \tau \rangle \rangle$.

Lemma 2 *The map*

$$\begin{array}{rcl} K[\alpha]^m & \to & \hom_K(R, K) \\ v & \mapsto & \pi_{v,w} \end{array}$$

is a K-isomorphism.

Proof. Considering dimensions, it suffices to show that the map is injective. Let $v \in K[\alpha]^m$ be nonzero. It suffices to show that $\pi_{v,w}$ is not the zero map. Let $v[i] = \mu \neq 0$ for some $0 \leq i < m$. Then the defining condition of w guarantees that there exists $\nu \in K[\alpha]$ such that $\langle w, \mu\nu \rangle \neq 0$. Thus, $\pi_{v,w}(\nu\beta^i) \neq 0$. $\square$

The lemma implies that if $v \in K[\alpha]^m$ is chosen at random, then $\pi_{v,w}$ is a random element of $\hom_K(R, K)$.

Now we have everything we need to present our algorithm. Let $\sigma \in R$ be the element whose minimal polynomial over K we wish to compute. Let $1 \leq l \leq d$ be a bound on the degree of this polynomial.

$g \leftarrow 1 \in K[x]$
$\tau \leftarrow 1 \in R$

while $\tau \neq 0$ do
  choose $v \in K[\alpha]^m$ at random
  $v \leftarrow \tau \circ v$
  $c_i \leftarrow \langle w, \langle v, \sigma^i \rangle \rangle$ for $i = 0, \ldots, 2(l - \deg g) - 1$
  $g' \leftarrow \mathrm{BerMass}(c_0, \ldots, c_{2(l-\deg g)-1})$

  $g \leftarrow gg'$
  $\tau \leftarrow \tau g'(\sigma)$

output g

The following claims are easy to verify:

(1) upon termination, this algorithm outputs $\phi_{\sigma/K}$;

(2) using the algorithms described above for power projection, polynomial evaluation, and transposed multiplication, along with the Berlekamp/Massey algorithm, the algorithm has an expected running time of

$$O(M(d)l^{1/2} + ld)$$

operations in K.

Also, as for the case $R = K[\alpha]$, several optimizations are possible, and if $\phi_{\sigma/k}$ is known to be irreducible, a single iteration with $v = (1, 0, \dots, 0)^T$ and $w = (1, 0, \dots, 0)^T$ suffices.

References

[1] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *J. Algorithms*, 1:259–295, 1980.

[2] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *J. Assoc. Comput. Mach.*, 25:581–595, 1978.

[3] J. Gordon. Very simple method to find the minimal polynomial of an arbitrary non-zero element of a finite field. *Electronic Letters*, 12:663–664, 1976.

[4] E. Kaltofen. Challenges of symbolic computation: my favorite open problems. Preprint, 1998.

[5] M. Kaminski, D. G. Kirkpatrick, and N. H. Bshouty. Addition requirements for matrix and transposed matrix products. *Journal of Algorithms*, 9:354–364, 1988.

[6] J. Massey. Shift-register synthesis and BCH coding. *IEEE Trans. Inf. Theory*, IT-15:122–127, 1969.

[7] V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *Proc. Int. Symp. on Symbolic and Algebraic Comp.*, pages 14–21, 1991.

[8] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symbolic Comp.*, 17(5):371–391, 1994.

[9] V. Shoup. A new polynomial factorization algorithm and its implementation. *J. Symbolic Comp.*, 20(4):363–397, 1995.

[10] J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity*, 2:187–224, 1992.

[11] D. Wiedemann. Solving sparse linear systems over finite fields. *IEEE Trans. Inf. Theory*, IT-32:54–62, 1986.