# Deterministic Equation Solving over Finite Fields

Christiaan van de Woestijne
Mathematisch Instituut, Universiteit Leiden
P.O. Box 9512
2300 RA Leiden
The Netherlands
cvdwoest@math.LeidenUniv.nl

## ABSTRACT

Deterministic algorithms are presented for the efficient solution of diagonal homogeneous equations in many variables over finite fields. As auxiliary algorithms, it is shown how to compute a field generator that is an $n$th power, and how to write elements as sums of $n$th powers, for a given integer $n$. All these algorithms take polynomial time in $n$ and in the logarithm of the field size, and are practical as stated.

## Categories and Subject Descriptors

F.2.1 [**Numerical Algorithms and Problems**]: Computations in finite fields; G.4 [**Mathematical Software**]: Algorithm design and analysis; I.1.2 [**Algorithms**]: Algebraic algorithms; Analysis of algorithms

## General Terms

Algorithms, Design, Performance

## Keywords

Deterministic algorithms, Equation solving, Finite fields

## 1. INTRODUCTION

Currently known algorithms for solving equations over finite fields include:

- brute force search
- algorithms for factoring polynomials (see [2, 4, 16])
- Shanks' algorithm for taking square (and higher) roots (see [12], [1], [4])
- methods for multivariate equations based on the above (see Section 8)
- Schoof's algorithm for taking square roots in prime fields (see [10])

However, all of these are either probabilistic (barring a proof of the Generalised Riemann Hypothesis for some) or take more than polynomial time. It should be especially noted that taking square or higher roots cannot be done in deterministic polynomial time if the GRH is not assumed.

In the course of my Ph.D. project with Hendrik W. Lenstra, Jr., I have developed an algorithm for solving general diagonal homogeneous equations in many variables over finite fields, that is both deterministic and runs in polynomial time. More specifically, the equation it solves is the following, for a given finite field $\mathbb{F}$:

$$\sum_{i=1}^{n} a_i x_i^n = b \tag{1}$$

with $a_i$ and $b$ all nonzero elements of $\mathbb{F}$. The algorithm works for any characteristic and any $n$, although when $n^2 > |\mathbb{F}|$ no solution may exist. Taking $b = -1$, we find an algorithm that computes a sequence of $n$th powers summing to zero, with at most $n + 1$ terms.

The formulation of the algorithm is nontrivial and involves several steps that are interesting in their own right, all of which seem to be new. These include an extension of Shanks's algorithm and a method for computing an element $\alpha$ of $\mathbb{F}$ whose $n$th power generates $\mathbb{F}$ over its prime field; whereas the solution of the general problem makes essential use of the ability to write elements of $\mathbb{F}$ as sums of $n$th powers. This last reduction is based on a group-theoretic proof of part of the Chevalley-Warning Theorem by Dem′yanov ([5], also given in [7]).

The algorithm uses $\tilde{O}(n^3(\log|\mathbb{F}|)^2)$ operations in $\mathbb{F}$ to finish, and its practicality will be evident from the description. In Section 8 I compare its asymptotic complexity with that of a straightforward probabilistic algorithm.

In the special subcase of finding a sequence of $n$th powers in $\mathbb{F}$ that sum to zero, the complexity drops to a mere $\tilde{O}(n^2 \log|\mathbb{F}|)$ operations in $\mathbb{F}$, and I believe that this is optimal in regard to the dependence on $|\mathbb{F}|$.

Applications abound especially for the case of quadratic forms, since if $\operatorname{char}\mathbb{F}$ is not 2, every quadratic form can be brought into diagonal form. Thus it follows that one can compute a rational point on a conic or higher-dimensional quadric over $\mathbb{F}$ by a fast deterministic algorithm.

In another direction, one can use my algorithm to show that the problem of constructing an isomorphism between regular quadratic spaces (of dimension 2 or higher) over $\mathbb{F}$ has a deterministic reduction to taking a square root of the quotient of the discriminants.

Full proofs of all results in this note will appear in my Ph.D. thesis "Deterministic equation solving over finite fields" (Universiteit Leiden, 2005). A preliminary version is available for download on my homepage.


*ISSAC'05*, July 24–27, 2005, Beijing, China.

## 2. PRELIMINARIES

I note that all complexity bounds given in this paper are "soft" bounds; this means that logarithmic factors have been ignored. See [16, Section 25.7] for a precise definition.

I assume that finite fields are given by means of an irreducible polynomial of the right degree over the prime field. I will denote by $\mathbb{F}$ a finite field with characteristic $p$ and having $q = p^e$ elements. For algorithms for the basic field operations, see [9]. I use the following facts about finite fields.

First, the multiplicative group $\mathbb{F}^*$ is cyclic of order $q-1$. Hence for a positive integer $n$ the quotient of $\mathbb{F}^*$ by its subgroup of $n$th powers has $d$ elements, where $d$ equals the g.c.d. of $n$ and the group size $q-1$. It follows that every $d$th power is also an $n$th power, and conversely, and we might just as well replace $n$ with $d$. If $x$ and $y$ are integers such that $xn + y(q-1) = d$, then for every $a \in \mathbb{F}^*$, we have $a^d = (a^x)^n$, hence conversion back and forth is easy. Therefore I will assume throughout the paper, when dealing with $n$th powers, that $n$ divides $q-1$.

Next, I give some facts about the sums of $n$th powers in $\mathbb{F}$.

**Proposition 2.1** *Let $\mathbb{F}$ be a finite field of $q$ elements, let $n$ be a positive integer and let $K$ be the subset of sums of $n$th powers of elements of $\mathbb{F}$. Then:*

*(i) $K$ is a subfield of $\mathbb{F}$.*
*(ii) If $K$ is a proper subfield of $\mathbb{F}$, then we have $n^2 > q$.*
*(iii) $K$ is equal to $\mathbb{F}$ if, and only if, $\mathbb{F}$ can be generated as a field by adjunction of an $n$th power to its prime field.*
*(iv) Every nonzero element of $K$ is a sum of at most $n$ $n$th powers.*

I refer to [7, Section 4.2] for proofs of these statements, which are due in essence to Tornheim [15].

Finally, I comment on the solvability of (1); I recall a few definitions. A form is called *isotropic* if it has a nontrivial zero, *anisotropic* if it has only the trivial zero, and *universal* if it represents every nonzero element. A form is said to *represent zero* if and only if it has a *nontrivial* zero.

Recall that if a form $f$ represents a nonzero element $b$, then it also represents all elements of the form $bx^n$ for some $x \in \mathbb{F}$. In this case also the coset of $b$ modulo the $n$th powers is said to be represented by the form $f$.

**Theorem 2.2** *Let $\mathbb{F}$ be a finite field of $q$ elements, and let $n$ be a positive integer. Then:*

*(i) Every diagonal form of degree $n$ over $\mathbb{F}$ in $n+1$ variables is isotropic.*
*(ii) Assume every element in $\mathbb{F}$ is a sum of $n$th powers in $\mathbb{F}$. Then every diagonal form of degree $n$ over $\mathbb{F}$ in $n$ variables with nonzero coefficients is universal.*

The first statement of the Theorem is part of the Chevalley-Warning theorem. Note that the Theorem is not at all sharp; a straightforward application of Weil's bounds on the number of solutions of diagonal equations over finite fields gives much stronger results (see [17], [13]). For example, if $q > n^4$, then every equation of the form $ax^n + by^n = c$ is solvable, if $abc \neq 0$.

The interest, then, of the Theorem lies in its possessing a *constructive* proof. For the first statement, this proof is due to Dem′yanov (see [5] and [7, Théorème 4.1]), and independently to Kneser for the case $n = 2$ (see [8, Theorem XI.4.4]). I was able to prove the second statement by an extension of the same method. This statement was first proved by Schwarz [11] on the stronger assumption that $d = (n, q-1) < p$, where $p$ is the characteristic of $\mathbb{F}$.

The proof depends on the following Proposition.

**Proposition 2.3** *Let $\mathbb{F}$ be a finite field of $q$ elements, and let $f = a_0X_0^n + \ldots + a_vX_v^n$ be a diagonal form of degree $n$ in $v+1$ variables over $\mathbb{F}$. Assume that $a_i \neq 0$ for $i = 0, \ldots, v$. Then:*

*(i) If $f$ is not isotropic, it represents at least $v+1$ distinct classes of $\mathbb{F}^*$ modulo $n$th powers.*
*(ii) If every element of $\mathbb{F}$ is a sum of $n$th powers and $f$ is not universal, then $f$ represents at least $v+1$ distinct classes of $\mathbb{F}^*$ modulo $n$th powers.*

I do not prove the Proposition here, for space considerations and also because my algorithms for solving (1) provide algorithmic proofs.

## 3. OUTLINE OF THE ALGORITHM

My algorithm for solving (1) is largely divided into three steps:

1. Given a finite field $\mathbb{F}$ and a positive integer $n$, compute $\alpha \in \mathbb{F}$ such that $\alpha^n$ generates $\mathbb{F}$ over its prime field. Thus, writing $e$ for the degree of $\mathbb{F}$ over $\mathbb{F}_p$, every element $b$ of $\mathbb{F}$ can be written as
$$b = \sum_{i=0}^{e-1} b_i(\alpha^i)^n, \tag{2}$$
with $b_i \in \mathbb{F}_p$.

2. Write elements of $\mathbb{F}$ as a sum of $n$th powers with at most $n$ terms. Thus, interpreting the $b_i$ just given as integers between 0 and $p-1$, inclusive, we have a sum of $n$th powers with $ep$ terms; reduce this to at most $n$ terms by a logarithmic decay of the number of terms. This solves (1) in the case where all coefficients are 1.

3. Writing $a_0 = -b$ and applying the second step to $-1, -a_0/a_1, \ldots, -a_0/a_n$, we find a system of equations of the form
$$\begin{cases} -a_0(y_{0,1}^n + \ldots + y_{0,h_0}^n) = 0 \\ -a_1(y_{1,1}^n + \ldots + y_{1,h_1}^n) = a_0x_{1,0}^n \\ -a_2(y_{2,1}^n + \ldots + y_{2,h_2}^n) = a_0x_{2,0}^n + a_1x_{2,1}^n \\ \qquad\vdots \qquad\qquad\qquad \vdots \\ -a_n(y_{n,1}^n + \ldots + y_{n,h_n}^n) = a_0x_{n,0}^n + \ldots + a_{n-1}x_{n,n-1}^n \end{cases} \tag{3}$$
Here initially, the right hand sides have only $x_{j,0}$ nonzero, and we have $h_j \leq n$ for $1 \leq j \leq n$, and $h_0 \leq n+1$. We now perform a reduction step that decreases the value of one of the $h_j$ by at least one, until one of them becomes 1, whereas in the meantime the $x_{j,i}$ in the right hand sides are filled in. The desired solution follows.

## 4. FINDING $n$TH POWER GENERATORS

I now give the first step of the algorithm in more detail.

Let $\mathbb{F}$ be a finite field and $n$ a positive integer. Proposition 2.1 tells us that if every element of $\mathbb{F}$ is an $n$th power, then

there exists some $\alpha$ in $\mathbb{F}$ such that $\alpha^n$ generates $\mathbb{F}$ over its prime field. In this Section I show that it is possible to compute such an element $\alpha$ efficiently (and deterministically). I use the following auxiliary result, which could be considered as a multiplicative version of the primitive element theorem for separable field extensions.

**Theorem 4.1** *There exists a deterministic algorithm that, given finite fields $K$ and $L$, with $K \subseteq L$, and nonzero elements $\alpha_1, \ldots, \alpha_t$ of $L$, computes integers $x_1, \ldots, x_t$ such that*

$$\alpha_1^{x_1} \cdots \alpha_t^{x_t} \text{ generates the field } K(\alpha_1, \ldots, \alpha_t) \text{ over } K,$$

*and uses $\tilde{O}((\log q) + te)$ operations in $L$, where $e = [L : K]$.*

The main result of this Section is as follows.

**Theorem 4.2** *There exists a deterministic algorithm which, given a finite field $\mathbb{F}$ of $q$ elements and a positive integer $n$ dividing $q-1$, computes $\beta \in \mathbb{F}$ such that $\beta^n$ generates $\mathbb{F}$ over its prime field, or correctly asserts that no such $\beta$ exists, using $\tilde{O}(n^2 + n \log q)$ operations in $\mathbb{F}$.*

The following sections outline the proofs of these results.

## 4.1 The compositum algorithm

Let $K$ be a finite field with $q$ elements, let $t$ be a positive integer, and for $i = 1, \ldots, t$, let $\alpha_i$ be algebraic over $K$ of degree $e_i$. I assume that all $\alpha_i$ are contained in some finite overfield $L$ of $K$, and hence the composite field $M = K(\alpha_1, \ldots, \alpha_t)$ is well defined. Its degree is equal to $\operatorname{lcm}(e_1, \ldots, e_t)$; I denote this degree by $g$. The notation $M$ for the composite field, and $g$ for its degree, is kept throughout this Section and the next.

In the course of the algorithms of this Section, we will have many occasions to compute the *degree* of an element $\alpha$ of a given finite field $L$ over a subfield $K$. The fastest way that I know for doing this is calling Algorithm 14.26 in [16], which the authors call the *iterated Frobenius* algorithm. This method computes degrees in $\tilde{O}(e)$ operations in $L$.

The compositum algorithm is based on the following key observation (cf. [3, Lemma 6.2]), in which $\phi$ denotes Euler's totient function. I do not prove this result here.

**Lemma 4.3** *Let $L/K$ be a finite cyclic extension of fields and let $b_1, \ldots, b_n$ be a basis for $L$ as a $K$-vector space. Then at least $\phi([L : K])$ of the basis elements generate $L$ over $K$ as a field. There exists a basis for $L$ over $K$ with exactly $\phi([L : K])$ field generators.*

*Remark.* Actually, as $\liminf_{n\to\infty} \dfrac{\phi(n) \log\log n}{n}$ exists and is positive (by Theorem 328 in [6]), it is to be expected that at least one in $c \log\log e$ elements of a basis for $L$ over $K$ is a field generator, where $c$ is a positive absolute constant.

*Algorithm.* It is now easy to formulate an algorithm that satisfies the conditions of Theorem 4.1 above: given $\alpha_1, \ldots, \alpha_t$, compute a basis for $K(\alpha_1, \ldots, \alpha_t)$ over $K$; then test all the basis elements for field generators. This testing can be done very efficiently as we already know the degrees of the $\alpha_i$. We must find at least one generator by virtue of Lemma 4.3.

## 4.2 Composing the right fields

I solve the problem of the computing the desired generator in three stages.

A prime field $\mathbb{F}_p$ is generated over itself by $1^n$; this gives the base step.

Now suppose the prime field is small, by which I mean $p \leq n$. Then we simply enumerate $n^2 + 1$ elements of $\mathbb{F}$ and compute their $n$th powers. Among these powers, there are at least $n + 1$ distinct elements; hence if we adjoin all of them to $\mathbb{F}_p$, we get a field with more than $n$ elements. The compositum algorithm given above now serves to merge these $n + 1$ powers into just one $n$th power that generates this subfield over $\mathbb{F}_p$.

A problem arises if $\mathbb{F}$ does not have $n^2 + 1$ elements. This is also the only case in which a solution to the current problem might not exist (cf. Proposition 2.1 above). However, because $n$ is so large, we can simply try every element of $\mathbb{F}$ and still use polynomial time in $n$ and $\log q$.

Next, suppose that there exists some subfield $K$ of $\mathbb{F}$ with $|K| > n$, for which we know an $n$th power generator $\gamma^n$. Let $\beta$ be the given generator for $\mathbb{F}$ over $\mathbb{F}_p$; we now have the following result.

**Lemma 4.4** *Suppose $|K| > n$, and let $c_0, \ldots, c_n$ be distinct elements of $K$. Then we have $(\beta + c_i)^n \notin K$ for at least one $i$ with $0 \leq i \leq n$.*

*Proof.* Assume the contrary. Then for all $i$, we have $(\beta + c_i)^{n|K|} = (\beta + c_i)^n$, so $(\beta + c_i)^{|K|-1}$ is an $n$th root of unity in $L$, of which there are at most $n$. By the pigeonhole principle, there exist $i$ and $j$ with $0 \leq i < j \leq n$ such that $(\beta + c_i)^{|K|-1} = (\beta + c_j)^{|K|-1}$, which implies that

$$\frac{\beta + c_i}{\beta + c_j} \in K.$$

But this is a contradiction, because $\beta$ is not in $K$ and $c_i \neq c_j$. □

**Corollary 4.5** *With the same assumptions as in the Lemma, the elements $(\beta + c_i)^n$ (for $i = 0, \ldots, n$) together generate $\mathbb{F}$ over $K$.*

*Proof.* Retaining the same elements $c_i$, apply the Lemma successively to all maximal subfields of $\mathbb{F}$ containing $K$. It follows that no such field contains all the elements $(\beta + c_i)^n$. Therefore these elements generate the whole field $\mathbb{F}$ over $K$. □

With a second call to the compositum algorithm, we "compose" $\gamma$ and the elements $\beta + c_i$ (for $i = 0, \ldots, n$) to find a single element $\alpha$ whose $n$th power generates $\mathbb{F}$ over $\mathbb{F}_p$. This solves our problem.

# 5. SELECTIVE ROOT EXTRACTION

In this Section I prove the following statement, which is needed for the second step of my main algorithm.

**Theorem 5.1** *There exists a deterministic algorithm which, given a finite field $\mathbb{F}$ with $q$ elements, a positive integer $n$,*

*and $n+1$ nonzero elements $a_0, \ldots, a_n$ of $\mathbb{F}$, determines integers $i$ and $j$ and an element $\beta \in \mathbb{F}$ such that $0 \le i < j \le n$ and*

$$a_i/a_j = \beta^n,$$

*using $\tilde{O}(n(\log q) + (\log q)^2)$ operations in $\mathbb{F}$.*

The Theorem says actually that, given $n+1$ elements in $\mathbb{F}$, we can compute an $n$th root of the quotient of two among them; however, without close analysis of the elements $a_i$ one cannot predict which two. Because of this selection feature, the method was called Selective Root Extraction. My algorithm is an extension of the Tonelli-Shanks algorithm for taking roots in cyclic groups, which we first discuss.

## 5.1 The Tonelli-Shanks algorithm

The essence of the Tonelli-Shanks algorithm was already given by Tonelli in 1891 [14] for the purpose of extracting square roots modulo primes of the form $4k+1$. It has subsequently been rediscovered by Shanks [12] and by Adleman, Manders, and Miller [1] in the 1970's, all of whom generalise the method to finding roots of arbitrary exponent, while Shanks also notes that the method can be applied to arbitrary cyclic groups. The discussion in [4, Section 1.5.1] is limited to the square root case.

A notable property of the Tonelli-Shanks algorithm is that, given an element whose order is large enough, it proceeds deterministically to compute roots of $a$. Generally, to use it we must first guess such an element (for example, a nonsquare if we want square roots). In the present application, we are able to determine elements of sufficiently high order without having to search for them.

A complexity analysis shows that, given an element of high enough order, the Tonelli-Shanks algorithm uses $O(n + (\log q)^2 \log n)$ operations in $\mathbb{F}$ to compute an $n$th root.

## 5.2 The Selective Root Algorithm

The Selective Root Algorithm is, in fact, the closest I can come to a deterministic root taking algorithm in finite fields, given the current state of knowledge. The main idea is simple.

Let $\mathbb{F}$ be a finite field, with $q$ elements, let $n$ be a positive integer dividing $q-1$, and let $a_0, a_1, \ldots, a_n$ be $n+1$ nonzero elements of $\mathbb{F}$. Denote by $G$ the subgroup of $\mathbb{F}^*$ generated by the $a_i$. The group $G$ is cyclic; therefore, the index $[G : G^n]$ is at most $n$. It follows that there exist $i$ and $j$ such that $a_i G^n = a_j G^n$; in other words, for this particular $i$ and $j$, there exists $\beta \in G$ with $\beta^n = a_i/a_j$.

Now $a_i$ and $a_j$ are such that their quotient has not too large order in $G$; on the other hand, the order of $\beta$ is rather large. Thus the real task of the algorithm is to look both for elements of large and of small order in the group $G$.

The actual algorithm that comes out of this first factors $n$ into primes and works with one prime at a time. This has the additional advantage, besides being simpler to understand, that the number of $a_i$s to be examined is at least halved with every processed prime.

The number of field operations performed by the algorithm is quadratic in $\log q$, and thus the algorithm has essentially cubic bit complexity. I have been unable to obtain an essentially quadratic bound, except in situations where the orders of the arguments $a_i$ are bounded (see Section 6.2 for an example).

The complexity would improve if we could replace the Tonelli-Shanks algorithm by a faster root taking algorithm. Now there do exist essentially quadratic probabilistic algorithms for root taking, which are mostly guises of Berlekamp's polynomial factorisation algorithm (see [2] or [16, Section 14.5], for example); but these do not seem to suit the present deterministic application.

# 6. SUMS OF LIKE POWERS

I now detail the second step of the main algorithm.

We consider a finite field $\mathbb{F}$. Given a positive integer $n$, can we write any given element of $\mathbb{F}$ as a sum of $n$th powers of elements of $\mathbb{F}$? And if so, how many such powers are needed?

This problem, known as Waring's problem for finite fields by analogy to its classical formulation with respect to the integers, has known active research in the 20th century. Some elementary results are recalled in Proposition 2.1 (especially (iv)), but these are not optimal in the cases where $n$ is small with respect to $|\mathbb{F}|$. A survey of recent results can be found in [18].

What concerns us here is the question of *actually computing* representations of given elements as sums of $n$th powers. My result is the following.

**Theorem 6.1** *There exists a deterministic algorithm which, given a finite field $\mathbb{F}$ with $q$ elements, a positive integer $n$ and a nonzero element $b$ of $\mathbb{F}$, determines elements $x_1, \ldots, x_n$ of $\mathbb{F}$ such that*

$$b = \sum_{i=1}^{n} x_i^n, \tag{4}$$

*or correctly asserts that $b$ is not a sum of $n$th powers in $\mathbb{F}$, using $\tilde{O}(n^2(\log q) + n(\log q)^2)$ operations in $\mathbb{F}$.*

As far as I know, this is the first efficient deterministic algorithm to write finite field elements as sums of powers.

The result below treats the special case of writing 0 as a sum of powers, in which case I have obtained an algorithm that uses only a linear amount of field operations.

**Theorem 6.2** *There exists a deterministic algorithm which, given a finite field $\mathbb{F}$ with $q$ elements and a positive integer $n$, determines elements $x_0, \ldots, x_n$ of $\mathbb{F}$ such that*

$$\sum_{i=0}^{n} x_i^n = 0, \tag{5}$$

*using $\tilde{O}(n^2 \log q)$ operations in $\mathbb{F}$.*

As already shown in the outline of the main algorithm, representing a nonzero element $a$ of $\mathbb{F}$ as a sum of $n$th powers is easy, given an $n$th power generator of $\mathbb{F}$ over its prime field. (If such a generator does not exist, then by Proposition 2.1 we must have $n^2 > q$, and we can simply enumerate all sums of $n$th powers in $\mathbb{F}$ to see if $a$ is among them.) We must, however, reduce the number of terms in the sum to at most $n$, as claimed in the Theorem.

## 6.1 The reduction algorithm

Thus, suppose that we have a representation as follows:

$$a = \sum_{i=1}^{M} y_i^n \tag{6}$$

for some positive integer $M$ and some nonzero elements $y_i$ of $\mathbb{F}$. If this representation is obtained from the form (2) by simply writing all the coefficients as sums of ones, we have $M \leq ep$, where $e = [\mathbb{F} : \mathbb{F}_p]$.

Now divide the sequence $M$ terms $y_i$ into $n+1$ subsequences, each having roughly the same number of components. Next, form the $n+1$ sums $S_0$ up to $S_n$, where $S_j$ is the sum of $y_i^n$ for all $y_i$ contained in the first $j+1$ subsequences.

If any of the $S_j$ is zero, we immediately discard all the corresponding terms from the sum (6). If not, we apply Selective Root Extraction (see above) to the sequence $(S_0, \ldots, S_n)$ and obtain

$$S_l = \beta^n S_k$$

for some $\beta \in \mathbb{F}$ and some integers $k$ and $l$ with $0 \leq k < l \leq n$. Therefore, if we multiply all terms in the first $k+1$ subsequences by $\beta$ and discard all terms in the $(k+2)$th up to $(l+1)$th sequences, the value of the sum (6) does not change.

In both cases, the number of terms $M$ will drop by a factor of about $\frac{n+1}{n}$, or more if more than one subsequence can be discarded. The trick is applicable as long as we have $M \geq n+1$; hence we will end up having $M \leq n$, as desired, and the number of iterations will be logarithmic in $q$.

I point out that attention is required for the internal representation of a sum of the form (6). Namely, the initial number of terms $M$ is exponential in $\log p$; hence at every iteration we would need to perform an exponential amount of exponentiations and summations. But taking advantage of the way in which this sum arises, we see that we can do much better: internally, we still remember the coefficients $b_i$ from (2) and keep track of how they are split up by the grouping into subsequences.

However, to get the complexity bound given in Theorem 6.1, one first brings the size of the coefficients $b_i$ down to at most $n$ at the cost of increasing their number somewhat, with the result that we need $O(\log \log q)$ iterations instead of $O(\log q)$.

For example, one can try to write the $b_i$ as sums of $n$th powers in $\mathbb{Z}$ by repeatedly subtracting the greatest possible $n$th power. This very quickly brings the $b_i$ down to about $n^n$, which is sufficient.

## 6.2 A special case

I now treat the special case of writing 0 as a sum of $n$th powers. For this it is enough to represent $-1$ in this way and bringing it to the other side of the equation; however, by a more involved approach we can improve the complexity of the algorithm significantly.

The reason that the algorithm of the last section has essentially cubic bit complexity is the use of the Tonelli-Shanks algorithm. A more precise consideration of this algorithm shows that the complexity is only cubic if the order of the element whose root we compute is divisible by a high power of some prime that also enters into $n$.

However, if $r$ is a prime, dividing $n$, that divides the order of $a$ to a higher order than it divides $n$, then we can use this to construct an element $\eta$ whose $n$th power is an $r$th root of unity. As is well known, if $\zeta$ is an $r$th root of unity, then $1 + \zeta + \ldots + \zeta^{r-1} = 0$; thus, if $r \leq n+1$, we solve our problem by summing powers of $\eta^n$.

In fact, we need not limit ourselves to primes dividing $n$; it is enough if $r$ divides $q-1$. As an example, if $n$ is odd, we can use $r = 2$; any element of even order is enough to apply our side exit; but we know such an element already, namely, $-1$. We thus write $1^n + (-1)^n = 0$.

Now we can modify my algorithm to take advantage of this situation: before applying Selective Root Extraction, we examine the element we want to take a root of, and if its order contains enough factors of some prime below $n+1$, we solve our problem by means of roots of unity. On the other hand, if this never happens (e.g., in the case when $q-1$ itself contains no high powers) we only apply the Tonelli-Shanks algorithm to elements of bounded order. This leads to a complexity of $\tilde{O}(n^2 \log q)$ operations in $\mathbb{F}$ in all cases.

Unfortunately, it seems very difficult to extend this technique to the case where we have distinct coefficients.

# 7. THE TRAPEZIUM METHOD

I now give details for the third and final step of the main algorithm. The following theorem is the main result of this note.

**Theorem 7.1** *There exists a deterministic algorithm which, given a finite field $\mathbb{F}$ with $q$ elements, a positive integer $n$ dividing $q-1$, and nonzero elements $a_1, \ldots, a_n$ and $b$ of $\mathbb{F}$, computes elements $x_1, \ldots, x_n$ of $\mathbb{F}$, such that*

$$\sum_{i=1}^{n} a_i x_i^n = b, \tag{7}$$

*or decides that such elements do not exist, using $\tilde{O}(n^3(\log q) + n^2(\log q)^2)$ operations in $\mathbb{F}$ to complete.*

The proof of this result was already embarked upon in the outline of the algorithm, Section 3.

As in the previous two steps, we might encounter insolubility of our problem only if $n^2 > q$, by Proposition 2.1. And again, this implies that we have enough time to enumerate all elements represented by the form $\sum_{i=1}^{n} a_i x_i^n$ over $\mathbb{F}$ and see if $b$ is among them. Therefore, we can assume henceforth that a solution exists.

Write $a_0 = -b$, and assume we built the system of equations (3) (repeated here for the convenience of the reader):

$$\begin{cases} -a_0(y_{0,1}^n + \ldots + y_{0,h_0}^n) = 0 \\ -a_1(y_{1,1}^n + \ldots + y_{1,h_1}^n) = a_0 x_{1,0}^n \\ -a_2(y_{2,1}^n + \ldots + y_{2,h_2}^n) = a_0 x_{2,0}^n + a_1 x_{2,1}^n \\ \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ -a_n(y_{n,1}^n + \ldots + y_{n,h_n}^n) = a_0 x_{n,0}^n + \ldots + a_{n-1} x_{n,n-1}^n \end{cases}$$

(The name "trapezium method" derives from the shape of this system of equations.)

The first equation is formed by writing $-1$ as a sum of $n$th powers, and then bringing $-1$ to the left. The others result from writing $-a_0/a_i$ as a sum of $n$th powers, for $i = 1, \ldots, n$. Hence initially we have $h_0 \leq n = 1$ and $h_j \leq n$ for $j = 1, \ldots, n$ by Theorem 6.1, whereas all $x_{j,i}$ are zero except the $x_{j,0}$ for $j = 1, \ldots, n$, which are all nonzero and remain so throughout the algorithm (this is very important).

The goal is to lower the $h_j$ until one of them becomes 1; then our problem is solved by moving the term $a_0 x_{j,0}^n$ to the left and dividing through by $x_{j,0}^n$.

We try to lower the $h_j$ by bringing the last term $a_j y_{j,h_j}^n$ to the other side. We get the sequence

$$\left(a_0 y_{0,h_0}^n\,,\ a_0 x_{1,0}^n + a_1 y_{1,h_1}^n\,,\ \ldots,\right.$$
$$\left. a_0 x_{n,0}^n + \ldots + a_{n-1} x_{n,n-1}^n + a_n y_{n,h_n}^n\right).$$

This sequence has $n+1$ elements, say $c_0, \ldots, c_n$. If one is zero, we are done anyway!

Otherwise, use Selective Root Extraction to compute $\beta \in \mathbb{F}^*$ with

$$\beta^n = c_k/c_l, \quad \text{i.e.} \quad c_k = \beta^n c_l$$

(assume $k > l$).

Replace now the $k$th term in the sequence by $\beta^n$ times the $l$th term, and we can reduce $h_k$ by one!

Thus, in at most $n^2$ steps, we will get one of the $h_j$ down to one. The complexity bound follows directly from the ones given above.

# 8. PROBABILISTIC METHODS

It is interesting to compare the running time achievements of the algorithms given in this note with probabilistic methods for solving the same problems. Let us consider (7), as being the most difficult to solve; assume $n$ divides $q-1$.

An obvious idea that comes to mind is the following: let $x_1, \ldots, x_{n-1}$ be random elements of $\mathbb{F}$, test whether

$$\left(\sum_{i=1}^{n-1} a_i x_i^n\right)/b$$

is an $n$th power in $\mathbb{F}$, and if it is, take its $n$th root by means of a probabilistic root taking method. (It is possible that better methods exist, but I do not know of any.)

Now for this to work, we must be sure that there are enough solutions, otherwise we are not likely to find one by guessing. Now to every $n-1$-tuple $(x_1, \ldots, x_n)$ corresponds either zero, one, or $n$ solutions to (7). A lower bound for the number of "lucky" elements of $\mathbb{F}^{n-1}$ is thus obtained by dividing the number of solutions to (7) by $n$.

From Weil's estimates given in [17], on page 502, we can prove that if $q > 3n^2$, there are at least $q^{n-1}/2$ representations of $b$ of the form (7). Thus if $q > 3n^2$, we may expect that every $2n$th element of $\mathbb{F}^{n-1}$ will give rise to at least one solution of (7).

It is easy to estimate the expected running time of this algorithm as

$$\tilde{O}(n^2 + n \log q)$$

operations in $\mathbb{F}$; here the computation of $n$th roots is done by polynomial factorisation, and takes $\tilde{O}(n \log q)$ operations by [16, Corollary 14.16]. (One should *not* use the Tonelli-Shanks algorithm.) I assume here that the generation of a random element of $\mathbb{F}$ is about as complex as multiplication.

Thus, a probabilistic method will be faster than my method except in the special case given in Section 6.2. However, this running time rapidly deteriorates if $q$ gets below $3n^2$, whereas my bounds remain valid for $q > n^2$.

For the case where $q < n^2$, Weil's lower bounds on the number of solutions become negative and brute force, made polynomial time with the aid of dynamical programming, is the only remaining method.

# 9. REFERENCES


[1] Leonard Adleman, Kenneth Manders, and Gary Miller. On taking roots in finite fields. In *18th Annual Symposium on Foundations of Computer Science (Providence, R.I., 1977)*, pages 175–178. IEEE Comput. Sci., Long Beach, Calif., 1977.

[2] Eric Bach. A note on square roots in finite fields. *IEEE Trans. Inform. Theory*, 36(6):1494–1498, 1990.

[3] Eric Bach, Joachim von zur Gathen, and Hendrik W. Lenstra, Jr. Factoring polynomials over special finite fields. *Finite Fields Appl.*, 7(1):5–28, 2001. Dedicated to Professor Chao Ko on the occasion of his 90th birthday.

[4] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[5] V. B. Dem′yanov. On representation of a zero of forms of the form $\sum_{i=1}^m a_i x_i^n$. *Dokl. Akad. Nauk SSSR (N.S.)*, 105:203–205, 1955.

[6] G.H. Hardy and E.M. Wright. *An introduction to the theory of numbers*. Oxford, at the Clarendon Press, 1965. Fourth edition, 3rd corrected printing.

[7] Jean-René Joly. Équations et variétés algébriques sur un corps fini. *Enseignement Math. (2)*, 19:1–117, 1973.

[8] T. Y. Lam. *The algebraic theory of quadratic forms*. W. A. Benjamin, Inc., Reading, Mass., 1973. Mathematics Lecture Note Series.

[9] H. W. Lenstra, Jr. Finding isomorphisms between finite fields. *Math. Comp.*, 56(193):329–347, 1991.

[10] René Schoof. Elliptic curves over finite fields and the computation of square roots mod $p$. *Math. Comp.*, 44(170):483–494, 1985.

[11] Štefan Schwarz. On universal forms in finite fields. *Časopis Pěst. Mat. Fys.*, 75:45–50, 1950.

[12] Daniel Shanks. Five number-theoretic algorithms. In *Proceedings of the Second Manitoba Conference on Numerical Mathematics (Univ. Manitoba, Winnipeg, Man., 1972)*, pages 51–70. Congressus Numerantium, No. VII, Winnipeg, Man., 1973. Utilitas Math.

[13] Charles Small. Diagonal equations over large finite fields. *Canad. J. Math.*, 36(2):249–262, 1984.

[14] Alberto Tonelli. Bemerkung über die Auflösung quadratischer Congruenzen. *Nachr. Göttingen*, (10):344–346, 1891. Reported in Dickson's History, Vol. 1, Ch. VII, item 193, p. 215.

[15] Leonard Tornheim. Sums of $n$-th powers in fields of prime characteristic. *Duke Math. J.*, 4:359–362, 1938.

[16] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, second edition, 2003.

[17] André Weil. Numbers of solutions of equations in finite fields. *Bull. Amer. Math. Soc.*, 55:497–508, 1949.

[18] Arne Winterhof. On Waring's problem in finite fields. *Acta Arith.*, 87(2):171–177, 1998.