Low Complexity Normal Elements over Finite Fields of Characteristic Two

ARIANE M. MASUDA, LUCIA MOURA, DANIEL PANARIO AND DAVID THOMSON

***Abstract*—In this paper we extend previously known results on the complexities of normal elements. Using algorithms that exhaustively test field elements, we are able to provide the distribution of the complexity of normal elements for binary fields with degree extensions up to 39. We also provide current results on the smallest known complexity for the remaining degree extensions up to 512 by using a combination of constructive theorems and known exact values. We give an algorithm to exhaustively search field elements by using Gray codes which allows us to reuse previous computations. We compare this with a standard method. We analyze this algorithm and show both experimentally and asymptotically that the Gray code optimization gives substantial savings. The total computation of the distribution of the complexity of normal elements for degrees up to 39 in our experiments allows us to draw several conjectures. In particular, our data provides remarkable evidence for the conjecture that the complexity of normal elements follows a normal distribution. Finally, we conjecture that there is no linear bound on the minimum complexity with respect to the degree of the extension.**

***Index Terms*—Finite fields, normal elements, low complexity, Gray codes.**

I. INTRODUCTION

LET $\mathbb{F}_q$ be a finite field of order q, and let $\mathbb{F}_{q^n}$ be a finite extension of $\mathbb{F}_q$. A *normal basis* of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ is a basis of the form $N = \{\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}\}$ where $\alpha \in \mathbb{F}_{q^n}$. In this case, we say that α is a *normal element* of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, or that α generates the normal basis N. The elements in N are called the *conjugates* of α. Normal bases exist in any finite extension of a finite field [14].

Let $\alpha_i = \alpha^{q^i}$ for $0 \le i \le n-1$, and let $T = (t_{ij})$ be the $n \times n$ matrix given by

$$\alpha\alpha_i = \sum_{j=0}^{n-1} t_{ij}\alpha_j, \quad 0 \le i \le n-1, \quad t_{ij} \in \mathbb{F}_q. \qquad (1)$$

The *complexity* of the normal basis N, denoted by c_N, is the number of non-zero entries in T. Mullin, Onyszchuk, Vanstone and Wilson [18] proved that $c_N \ge 2n-1$. The normal basis N is said to be *optimal* when this lower bound is achieved, that is, when $c_N = 2n-1$. Optimal normal elements exist but not in all finite fields (for instance, see [15, Chapter 3]). Optimal normal bases over finite fields were completely characterized by Gao and Lenstra [8].

The implementation of finite field arithmetic is highly required in several applications such as coding theory, cryptography and signal processing; see for example [16]. In particular, an important operation for these applications is exponentiation of elements in a finite field. It is well known that the use of normal bases yields efficient exponentiation in $\mathbb{F}_{2^n}$ [7], [23], including applications to hardware implementations [20]. Specifically, exponentiations to 2^k, for any positive integer k, are just cyclic bit shifts. Moreover, when using normal bases the speed of multiplications over $\mathbb{F}_{2^n}$ depends directly on c_N (see [5, Section 11.2.2]). Thus, it is important to use a normal basis in $\mathbb{F}_{2^n}$, for any given n, with the lowest possible complexity.

When no optimal normal basis exists, the problem of classifying *low complexity* normal bases still remains open. There is no proper definition of the term "low complexity". Ideally, one expects to have a complexity bounded by kn for some small constant k. Young and Panario [26] conjecture that low complexity normal elements over finite fields of characteristic two with complexity up to $3n$ only occur in finite fields with an optimal normal element. Wan and Zhou [25] extended part of the results in [26] for finite fields of odd characteristic.

For a better understanding of the behavior of the complexities of normal elements, tables summarizing the complexity distribution are important tools. In [15, Section 3.3], Jungnickel provides a table with minimum and maximum complexities of normal elements in $\mathbb{F}_{2^n}$, for each $n \le 30$. In [15, Section 5.4], for $31 \le n \le 60$, he provides a table due to Geiselmann [13] with the lowest complexities found via free polynomials. In this paper, we present the distribution of the complexity of normal elements for $n \le 39$. We gather, for each n in this range, the frequency of each possible complexity value. We give summary tables with minimum, maximum, average and variance values for the complexities of normal elements and self-dual normal elements in $\mathbb{F}_{2^n}$. We do a statistical fit of the frequency data, for each $n \le 39$, that clearly suggests that the complexities of normal elements in $\mathbb{F}_{2^n}$ follow a normal distribution. Our data is obtained via an exhaustive search in $\mathbb{F}_{2^n}$ which is substantially sped up by visiting the finite field elements in a Gray code order. We also extend the table of the smallest known complexities for other practical values of interest ($40 \le n \le 512$), using a combination of previously known results and random search.

This paper is organized as follows. In Section II, we give a brief survey of known results about normal elements that are relevant and used in our research. In Section III, we describe

Ariane M. Masuda is with the Department of Mathematics and Statistics, University of Ottawa, amasuda@uottawa.ca†.

Lucia Moura is with the School of Information Technology and Engineering, University of Ottawa, lucia@site.uottawa.ca.

Daniel Panario is with the School of Mathematics and Statistics, Carleton University, daniel@math.carleton.ca.

David Thomson is with the Department of Electrical and Computer Engineering, University of Waterloo, d2thomso@uwaterloo.ca†.

† Ariane M. Masuda and David Thomson were at the School of Mathematics and Statistics, Carleton University, when this research was performed.

Ariane M. Masuda, Lucia Moura and Daniel Panario are supported in part by NSERC of Canada.

and analyze the algorithms that we use for the exhaustive complexity computations. Our asymptotic analysis shows that using a Gray code order, instead of an arbitrary order, allows a 21.05% reduction on the running time. Tables and conjectures are given in Section IV.

II. Previous results

We start with a criterion, due to Davenport, that we use to determine whether an element is normal or not.

Theorem 1: [15, Theorem 3.1.8] Let α be an element in $\mathbb{F}_{q^n}$. Then α is a normal element of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ if and only if the polynomials $x^n - 1$ and

$$\alpha^{q^{n-1}} x^{n-1} + \cdots + \alpha^q x + \alpha$$

in $\mathbb{F}_{q^n}[x]$ are relatively prime.

The number of normal elements was established by Ore.

Theorem 2: [15, Theorem 3.1.5] Let q be a power of the prime p, let n be a positive integer and write $n = p^a m$, where p does not divide m. Then the number of normal bases of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ equals

$$\frac{1}{n}\Phi_q(x^n - 1) = \frac{q^n}{n}\prod_{d|m}(1 - q^{-o_d(q)})^{\phi(d)/o_d(q)},$$

where $\Phi_q(f)$ is the number of polynomials in $\mathbb{F}_q[x]$ of degree smaller than the degree of f which are relatively prime to f, $o_n(a)$ is the order of a modulo n, and $\phi(d)$ is the number of positive integers smaller than d that are relatively prime to d.

Several authors have provided lower and upper bounds for the number of normal bases of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. Here, we present Gao and Panario's upper bound [9, Theorem 3.4]:

$$\frac{1}{n}\Phi_q(x^n - 1) \leq \frac{q^n}{e^{\gamma - c_q} n\sqrt{1 + \log_q n}}, \tag{2}$$

where $\gamma = 0.577216\ldots$ is Euler's constant and $c_q = q/\left((q-1)(\sqrt{q}-1)\right)$; see also [6]. Von zur Gathen and Giesbrecht [12] showed that the probability that an arbitrary element in $\mathbb{F}_{q^n}$ forms a normal basis is larger than $1/(16\log_q n)$.

Specific criteria to identify optimal normal elements are provided by Mullin, Onyszchuk, Vanstone and Wilson [18]. As a consequence of these criteria, given n, it is simple to determine whether there exists an optimal normal basis in $\mathbb{F}_{q^n}$ or not. When $q = 2$, a list with all such values of n up to 1300 can be found in [15, Table 3.1]. There is also a list of all complexities of normal bases for $\mathbb{F}_{2^n}$ with n up to 30 in [15, Table 3.2]. Using our algorithms, we extend this table for n up to 39 (Table IV). For n from 40 to 512, we give the smallest complexity found using different methods from the literature (Table V). The details will be explained later; we now present some results that are used to compute these tables.

We present a theorem using generalized Gauss periods to construct normal bases with known complexities. This theorem was shown for extensions $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ by Ash, Blake and Vanstone [1], and for general q independently by Beth, Geiselmann and Meyer [2].

Theorem 3: [1], [2] Let q be a prime or a prime power, and let n and k be positive integers such that $nk+1$ is a prime not dividing q. Let β be a primitive $(nk+1)$th root of unity in $\mathbb{F}_{q^{nk}}$. Suppose that $\gcd(nk/e, n) = 1$ where e is the order of q modulo $nk+1$. Then, for any primitive kth root of unity τ in $\mathbb{Z}_{nk+1}$, the element

$$\alpha = \sum_{i=0}^{k-1} \beta^{\tau^i}$$

generates a normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ with complexity at most $(k+1)n - k$, and at most $kn - 1$ if $k \equiv 0 \pmod{p}$, where p is the characteristic of $\mathbb{F}_q$.

This construction defines optimal normal elements for $k = 1$ for all q, and for $k = 2$ and $q = 2$. These elements generate the so-called Type I and Type II optimal normal bases, respectively. In [8], Gao and Lenstra prove that these elements characterize all optimal normal bases over finite fields. In general, the exact complexity of a basis constructed using Theorem 3 is difficult to analyze. For $q = 2$ we have the following special cases.

Theorem 4: [1] Let $q = 2$. Then the normal basis N generated by α as constructed in Theorem 3 has complexity

$$c_N = \begin{cases} 4n - 7 & \text{if } k = 3, 4 \text{ and } n > 1; \\ 6n - 21 & \text{if } k = 5 \text{ and } n > 2 \text{ or } k = 6 \text{ and } n > 12; \\ 8n - 43 & \text{if } k = 7 \text{ and } n > 6. \end{cases}$$

The following recursive construction allows us to cover more finite fields in Table V.

Theorem 5: [15, Theorem 3.3.13] Let α and β generate normal bases A and B for $\mathbb{F}_{q^m}$ and $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, respectively. Assume that m and n are coprime and put $\gamma := \alpha\beta$. Then γ generates a normal basis N for $\mathbb{F}_{q^{mn}}$ over $\mathbb{F}_q$ and $c_N = c_A c_B$. Furthermore, if α and β both generate optimal normal bases, then γ has complexity $c_N = 4mn - 2m - 2n + 1$.

As a consequence (see [15, Corollary 3.3.15]), if we let the *complexity* of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ be

$$C_q(n) := \min\{c_N \colon N \text{ is a normal basis for } \mathbb{F}_{q^n} \text{ over } \mathbb{F}_q\},$$

and if m and n are relatively prime, then $C_q(mn) \leq C_q(m)C_q(n)$.

Next we give a class of theorems which allows us to construct even more low complexity normal bases in subfields of finite fields containing optimal normal bases. To do this, we first define the *trace* of an element $\alpha \in \mathbb{F}_{q^n}$ over $\mathbb{F}_{q^m}$, where $n = km$, by $\mathrm{Tr}_{q^n/q^m}(\alpha) = \alpha + \alpha^{q^m} + \cdots + \alpha^{q^{(k-1)m}}$. We present the statement of the theorem for the q even and Type I case, and give a table of results for both Type I and Type II cases (Table I). There are analogous results for the q odd case, and for the dual bases of the given constructions, however in this paper we focus only on finite fields of even characteristic.

Theorem 6: [4] Let $\alpha \in \mathbb{F}_{2^n}$ generate an optimal normal basis of Type I of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, $n > 2$, and let $\beta = \mathrm{Tr}_{2^n/2^m}(\alpha) \in \mathbb{F}_{2^m}$ with $m = n/k$ and $k \leq m$. Then, an upper bound for the complexity of the normal basis of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ generated by β is $(k+1)m - 3k + 2$ if m is even and k is odd, or $km - k + 1$ otherwise.

A table containing optimal normal elements, the number of normal bases, the minimum and the maximum complexity in

TABLE I
SUMMARY OF BEST-KNOWN LOW COMPLEXITIES FOR $\mathbb{F}_{2^m} \subseteq \mathbb{F}_{2^n}$ OBTAINED BY TRACES, WHERE $m = n/k$.

	Type I	Type II
m odd	$km - k + 1$	$2km - 2k + 1$
m even, k odd	$(k+1)m - 3k + 2$	for all m
m even, k even	$km - k + 1$	

$\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ for $n \leq 30$ was presented by Mullin, Onyszchuk, Vanstone and Wilson [18]. For $n \leq 27$, this information was obtained through a computer search. For $n = 28$, 29 and 30, no computer search was performed by these authors due to the computational complexity and the observation that these fields contain optimal normal bases.

We focus now on self-dual normal bases. Self-dual normal bases form a special class of normal bases that have also been used in finite field implementations [24]. Suppose $A = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ is a basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. A *dual basis* $\{\beta_0, \beta_1, \ldots, \beta_{n-1}\}$ of A is defined, for $0 \leq i, j, \leq n$, by

$$\mathrm{Tr}(\alpha_i \beta_j) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

The basis A is called *self-dual* if $\beta_i = \alpha_i$ for all i, $0 \leq i \leq n-1$. It is well known that, for each basis $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ of $\mathbb{F}_{q^n}$, there exists a unique dual basis. Also, the dual of a normal basis is always normal. The following result provides a simple way of recognizing self-dual normal bases.

Theorem 7: [15, Corollary 5.1.3] Let α be a generator for a normal basis B of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, and $T = (t_{ij})$ be the matrix defined in (1). Then B is self-dual if and only if T is symmetric.

A self-dual normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ exists if and only if either q is even and n is not a multiple of 4 or both q and n are odd (see [15, Theorem 5.2.1]). As a consequence, there is no self dual normal basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ if 4 divides n. If $n \equiv 2 \pmod 4$, the next two results show how to obtain a self-dual normal basis over $\mathbb{F}_{2^n}$, provided that a basis of the same type is given, and how their complexities are related.

Theorem 8: [15, Corollary 5.4.3] Let $\alpha \in \mathbb{F}_{2^n}$, where n is even, and put $\gamma := 1+\alpha$. Then α generates a self-dual normal basis if and only if γ does.

Theorem 9: [15, Theorem 5.4.4] Let α generate a self-dual normal basis B for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, and assume that n is even. Put $\gamma := 1 + \alpha$, and let B^C be the self-dual normal basis generated by γ. Then the complexities of B and B^C are related as follows

$$C_{B^C} = n^2 - 3n + 8 - C_B.$$

For $n \equiv 2 \pmod 4$, the average complexity of a self-dual normal basis for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ is $\frac{1}{2}(n^2 - 3n + 8)$ (see [15, Corollary 5.4.5]). Also, if C_B is the complexity of a self-dual normal basis B then

$$2n - 1 \leq C_B \leq n^2 - 5n + 9.$$

Equality holds in one of these bounds if and only if either B or B^C is optimal; in this case, $2n + 1$ is prime and 2 is a primitive root modulo $2n + 1$ (see [15, Theorem 5.4.6]).

III. ALGORITHMS FOR COMPUTING THE COMPLEXITY DISTRIBUTION OF NORMAL ELEMENTS

In this section, we describe and analyze two variants of an exhaustive algorithm for computing the complexity of each normal element in $\mathbb{F}_{q^n}$. Algorithm `StandardNCD` (for "Standard Normal Complexity Distribution") is the simplest variant. Algorithm `GrayCodeNCD` (for "Gray Code Normal Complexity Distribution"), the second variant, uses Gray codes in order to efficiently update the current finite field element and its conjugates. We use basic operations in $\mathbb{F}_q$ as our time complexity measure. Then we show that asymptotically the Gray code variant reduces the time complexity of the standard algorithm by 21.05%. We present the algorithms and their analyses for $q = 2$. Their extensions to general q are straightforward.

A. Description of the algorithms

We now describe Algorithm `StandardNCD`; its pseudocode appears in Fig. 1. At each step, we compute $\alpha \in \mathbb{F}_{2^n}$, represented as a polynomial over $\mathbb{F}_2$ with coefficients stored in the tuple $\alpha_{\mathrm{coeff}} = [a_{n-1}, \ldots, a_0]$. We define variables $\alpha_j = \alpha^{2^j}, j = 0, 1, \ldots, n-1$, to store α and its conjugates. The main loop runs through $\alpha \in \mathbb{F}_{2^n}^*$, updating the tuple α_{coeff} with the next binary tuple, using an arbitrary order computed via the function `NextTuple()` in line 12.

Each field element α is processed in lines 6-11. Since conjugates form an equivalence class for the computation of the complexity of normal elements, we only compute the complexity for one canonical representative of the normal basis. We stipulate that α is canonical if and only if α is the (unique) minimum element among $\{\alpha^{2^j}\}_{j=0}^{n-1}$, when using the lexicographical ordering of the coefficients of their polynomial representations. For a canonical element α, we check if it is normal using the procedure `IsNormal` which employs a gcd test based on Theorem 1. If α is normal, then we calculate its complexity via the procedure `CalculateComplexity`, which is an implementation of the complexity definition; see (1). This amounts to solving an $n \times n$ system of equations with coefficients in $\mathbb{F}_2$. Since the only difference between these systems of equations are their right-hand sides, we calculate the inverse of the matrix P associated with the system and then apply n multiplications of P^{-1} by vectors in $\mathbb{F}_2^n$.

The values of α and its conjugates are updated in lines 12-15. Algorithm `StandardNCD` does a straightforward update, while Algorithm `GrayCodeNCD` employs a more efficient update that uses a Gray code.

A *Gray code* is an ordering of the 2^n binary vectors of length n such that any two consecutive vectors have Hamming distance equal to 1, i.e. any two consecutive vectors differ in a unique position k. Gray codes are well studied (see [21]) and they exist for every n. In particular, we can build a Gray code in which the first vector is the zero vector, and each next vector can be computed in time linear with n. In the `GrayCodeNCD` algorithm, this computation is done by calling the function `NextGrayTuple()` in line 12.

Now we describe how Gray codes are used to update α. Let α' be the field element in the previous iteration and

α be the field element in the current iteration, represented as polynomials over $\mathbb{F}_2$. Since their polynomial coefficients correspond to successive tuples in a Gray code, we have $\alpha = \alpha' + x^k$, for some $k \in \{0, 1, \ldots, n-1\}$. Therefore, $\alpha^{2^j} = (\alpha' + x^k)^{2^j} = (\alpha')^{2^j} + (x^k)^{2^j}$. In lines 1-2 of the algorithm `GrayCodeNCD`, the values $e_i^j = (x^i)^{2^j}$ are precomputed, and used in line 15 for updating α. This reduces the computation in lines 14-15 from $n-1$ squarings, as in the standard algorithm, to n additions in the Gray code variant.

We recall from Theorem 7 that a self-dual normal basis has a symmetric multiplication table. So, in order to adapt the given algorithms to search exclusively for self-dual normal elements, we alter the `CalculateComplexity` procedure to solve for t_{ij} row-by-row (i.e. solving the system up to n times), and check the non-diagonal elements (up to the index of the calculated row) for symmetry. If this follows to completion, the normal element is self-dual, and the complexity is stored. We call this algorithm `SelfDualNCD`.

B. *Analyses of the algorithms*

In this section, we analyze the amortized worst-case running time per finite field element, which we call $TS(n)$ and $TG(n)$, for standard and Gray code variants, respectively. Therefore, the total running time of each algorithm will be at most $2^n TS(n)$ and $2^n TG(n)$, respectively. We do our analysis in terms of $M(n)$, the number of basic operations in $\mathbb{F}_2$ used in a multiplication of polynomials of degree smaller than n over $\mathbb{F}_2$. We then compute the savings in running time given by the Gray code variant with respect to the standard one.

For the multiplication of polynomials of degree n over $\mathbb{F}_2$, we consider the classical method in which $M(n) = 2n^2 + O(n)$ (see Section 2.3 in [10]), and Karatsuba's method in which $M(n) \leq 9n^{\log_2 3} \leq n^{1.59}$ (see Section 8.1 in [10]). In addition, we observe that a multiplication in $\mathbb{F}_{2^n}$ corresponds to one multiplication of polynomials of degree at most $n-1$ followed by a modular reduction by the polynomial defining the extension $\mathbb{F}_{2^n}$. This multiplication in $\mathbb{F}_{2^n}$ costs $3M(n) + O(n)$ operations in $\mathbb{F}_2$ (see Section 9.7 in [10]).

We first analyze the steps that are common to both algorithms. The canonicity test in line 6 can be done in $O(n^2)$ comparisons in $\mathbb{F}_2$. Procedure `IsNormal` called in line 7 is only run for $1/n$ of the field elements. Each time it is run, it involves the computation of a gcd between polynomials of degrees $n-1$ and n over $\mathbb{F}_{2^n}$ (see Theorem 1). This gcd can be computed, using the Euclidean algorithm, in at most $n+1$ inversions and $2.5n^2 + O(n)$ additions and multiplications in $\mathbb{F}_{2^n}$ (see Section 3.3 in [10]). As we have seen, multiplications in $\mathbb{F}_{2^n}$ can be executed in $3M(n) + O(n)$ operations in $\mathbb{F}_2$, while additions require $O(n)$. Since $n+1$ inversions in $\mathbb{F}_{2^n}$ can be computed in $O(n^3)$ operations in $\mathbb{F}_2$, line 7 can be computed in time $7.5n^2 M(n) + O(n^3)$ operations in $\mathbb{F}_2$. Therefore, the amortized time for line 7 is $7.5nM(n) + O(n^2)$.

Lines 8-9 are only executed $\Phi_2(x^n - 1)/n$ times in total, since the number of normal elements is $\Phi_2(x^n - 1)$ by Theorem 2. Each time these lines are run, algorithm `CalculateComplexity` is executed. For this algorithm, steps 1-2 take $O(nM(n))$, steps 3-7 and 10 take $O(n^2)$ and steps 8-9 take $O(n^3)$. So, the running time of `CalculateComplexity` is $O(nM(n) + n^3) = O(n^3)$. Thus, the amortized cost for lines 8-9 in both main algorithms is $O(n^3 \Phi_2(x^n - 1)/(n2^n))$, which, by the upper bound in (2), is $O(n^2/\sqrt{\log n})$.

The iterations of each algorithm differ in the updates performed in lines 12-15. For the standard algorithm, the cost of these lines is dominated by $n-1$ squarings of an element in $\mathbb{F}_{2^n}$. Each of these operations can be done by squaring a polynomial over $\mathbb{F}_2$ of degree smaller than n followed by a modular reduction by a polynomial of degree n. The cost of each squaring in $\mathbb{F}_{2^n}$ is then dominated by the modular reduction at a cost of $2M(n) + O(n)$. Thus, lines 12-15 for the standard algorithm cost $2nM(n) + O(n^2)$ operations in $\mathbb{F}_2$. On the other hand, for the Gray code algorithm, the cost of lines 12-15 is dominated by n additions of polynomials of degree smaller than n over $\mathbb{F}_2$, which can be done in time $O(n^2)$.

Combining the above analyses for the steps of each algorithm, we get

$$\begin{aligned} TS(n) &= 7.5nM(n) + 2nM(n) + O(n^2) \\ &= 9.5nM(n) + O(n^2), \\ TG(n) &= 7.5nM(n) + O(n^2). \end{aligned}$$

Using the above equations, we conclude that the Gray code variant gives (asymptotically) savings of $(9.5 - 7.5)nM(n)/9.5nM(n) \approx 21.05\%$, independently of the multiplication method used.

Finally, in Table II, we give asymptotic estimates of $TS(n)$ and $TG(n)$ for specific values of $M(n)$, using classical and Karatsuba's multiplication methods. There are other multiplication methods with smaller asymptotic running times, but these methods are less efficient for smaller n's (see [11, Section 7]). The cross-over happens for n's much larger than the ranges our algorithms could be applied, since iterations are repeated 2^n times.

TABLE II
RUNNING TIMES (PER FIELD ELEMENT) USING DIFFERENT MULTIPLICATION METHODS.

Method	Karatsuba's	Classical
$M(n)$	$9n^{\log_2 3} \leq 9n^{1.59}$	$2n^2$
$TS(n)$	$85.5n^{1+\log_2 3} + O(n^2)$	$19n^3 + O(n^2)$
$TG(n)$	$67.5n^{1+\log_2 3} + O(n^2)$	$15n^3 + O(n^2)$

IV. RESULTS AND CONJECTURES

Table III gives a comparative runtime analysis of all the algorithms, and Table IV outlines the main results found in this experiment using the `GrayCodeNCD` and `SelfDualNCD` algorithms for $n \leq 39$. In Table V we give the best known complexities found for $40 \leq n \leq 512$ using several sources from the literature. These results are analyzed, and conjectures are presented based on the findings.

The experiments were performed on individual Pentium IV 3.0 GHz systems with 1.5 GB of DDR RAM. The operating

```
Algorithm StandardNCD($n$)
    $\alpha_{\rm coeff}$=NextTuple($[0,0,\ldots,0]$);
    $\alpha_0$ =polynomial($\alpha_{\rm coeff}$); /* initialize $\alpha$ */
    for $j=1$ to $n-1$ do
        $\alpha_j = (\alpha_{j-1})^2$; /* calculate conjugates $\alpha_j = (\alpha)^{2^j}$ */
    for $i=1$ to $2^n-1$ do /* run through $\mathbb{F}_{2^n}^*$ */
        if (($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$) is canonical) then
            if (IsNormal($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$)) then
                Compl=CalculateComplexity($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$);
                UpdateStats($\alpha_0$,Compl);
            endif
        endif
        $\alpha_{\rm coeff}$ =NextTuple($\alpha_{\rm coeff}$); /* go to next */
        $\alpha_0$ =polynomial($\alpha_{\rm coeff}$);
        for $j=1$ to $n-1$ do
            $\alpha_j = (\alpha_{j-1})^2$;/* calculate $\alpha_j = \alpha^{2^j}$ */
    endfor
    PrintFinalStats();
end StandardNCD

Algorithm GrayCodeNCD($n$)
    for $i=0$ to $n-1$ do $e_i^0 = x^i$; /* precomputation */
    for $j=1$ to $n-1$ do $e_i^j = (e_i^{j-1})^2$; /* precomputation */
    $\alpha_{\rm coeff}$=NextGrayTuple($[0,0,\ldots,0]) = [0,\ldots,0,1]$;
    for $j=0$ to $n-1$ do $\alpha_j = e_0^j$;/* initial. $\alpha$ & conjug.*/
    for $i=1$ to $2^n-1$ do /* run through $\mathbb{F}_{2^n}^*$ */
        if (($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$) is canonical) then
            if IsNormal($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$) then
                Compl=CalculateComplexity($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$);
                UpdateStats($\alpha_0$,Compl);
            endif
        endif
        $\alpha'_{\rm coeff} = \alpha_{\rm coeff}$; $\alpha_{\rm coeff}$ =NextGrayTuple($\alpha'_{\rm coeff}$); /* next */
        Let $k$ be the index $\ell$ such that $\alpha_{\rm coeff}[\ell] \neq \alpha'_{\rm coeff}[\ell]$;
        for $j=0$ to $n-1$ do
            $\alpha_j = \alpha_j + e_k^j$; /* efficient $\alpha_j$ computation */
    endfor
    PrintFinalStats();
end GrayCodeNCD


Procedure CalculateComplexity($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$)
    for $i=0$ to $n-1$ do
        $\beta_i = \alpha_0 * \alpha_i$;
    for $i=0$ to $n-1$ do
        for $j=0$ to $n-1$ do
            $P_{ij}$ = $j$-th coefficient of $\alpha_i$;
            $A_{ij}$ = $j$-th coefficient of $\beta_i$;
        endfor
    Calculate $P_{inv} = P^{-1}$;
    Calculate $T = A \times P_{inv}$; /* $T = (t_{ij})$ */
    Compl = $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} t_{ij}$;
    return Compl;
end CalculateComplexity


Procedure IsNormal($\alpha_0,\alpha_1,\ldots,\alpha_{n-1}$)
    if ($\gcd(x^n-1, \alpha_0+\alpha_1 x+\cdots+\alpha_{n-1}x^{n-1}) = 1$)
       then return true;
       else return false;
end IsNormal
```

Fig. 1. Algorithms for exhaustively searching and enumerating all normal bases of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$.

TABLE III
RUNNING TIME IN SECONDS FOR SEVERAL ALGORITHMS.

n	Standard	GrayCode	SelfDual
2	0.00	0.00	0.00
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.00	0.00	0.00
6	0.00	0.00	0.00
7	0.00	0.00	0.00
8	0.00	0.00	0.00
9	0.00	0.01	0.00
10	0.01	0.01	0.01
11	0.03	0.03	0.02
12	0.07	0.06	0.05
13	0.16	0.13	0.11
14	0.43	0.27	0.23
15	0.65	0.52	0.48
16	1.48	1.13	1.02
17	3.20	2.42	2.21
18	6.37	4.96	4.75
19	14.11	10.52	9.67
20	29.18	21.50	20.49
21	57.34	41.74	39.67
22	127.11	92.23	83.80
23	281.38	194.05	176.11
24	546.18	375.23	355.41
25	1157.33	833.03	741.50
26	2416.47	1672.35	1513.56
27	4983.14	3252.12	3011.80
28	9116.89	6776.99	7276.91
29	21179.50	14994.80	13115.90
30	40738.50	27515.20	25103.80

system was Red Hat Linux Enterprise Edition kernel 2.6.9-34.EL. We use C++, compiled using g++ 3.4.3, for all programming tasks. For arithmetic performed over $\mathbb{F}_2$, Shoup's NTL package version 5.3.2 was used [22], specifically taking advantage of the optimized binary arithmetic "GF2" libraries.

A. *Exhaustive search for* $n \leq 39$

Table III gives the CPU User-time for the `StandardNCD`, `GrayCodeNCD`, and the `SelfDualNCD` algorithms, measured in seconds. We observe from Table III that the running time of the `SelfDualNCD` algorithm is approximately ten percent faster than the `GrayCodeNCD` algorithm running time, though no analysis of this algorithm is presented. Moreover, a comparison of the running time per element shows an improvement in the `GrayCodeNCD` algorithm over the `StandardNCD` ranging from 26% to 35% for $20 \leq n \leq 30$. We also include Fig. 2 for graphical comparison.

Table IV shows findings on the complexity of normal elements for every finite field $\mathbb{F}_{2^n}$ with $n \leq 39$, computed with the implementation of the `GrayCodeNCD` algorithm. This includes the number of normal bases found $((\Phi_2(x^n-1))/n)$, the smallest and the largest complexities (m_{c_N}, M_{c_N}), average complexity, variance and standard deviation (Avg_{c_N}, Var_{c_N}, σ_{c_N}), and the smallest and the largest complexities for self-dual normal elements (m'_{c_N}, M'_{c_N}). Due to the time limitations, no search was performed for self-dual normal elements for $n \geq 37$. The optimal normal bases found are in agreement with the theorem of Gao and Lenstra [8] that characterizes for which fields optimal normal bases exist.

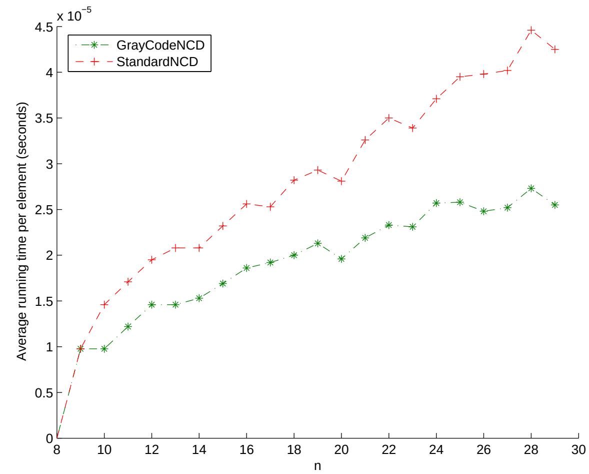


Fig. 2. Average runtime per element for two exhaustive algorithms.

B. *Lowest found complexities for* $40 \leq n \leq 512$

Table V is devoted to the cases when $40 \leq n \leq 512$. For these values of n, no self-dual testing has been performed. When an optimal normal element is known not to exist, we show the lowest complexity found by using known methods from the literature. In the first method considered, we check the conditions of Theorem 3 for values of k in the range established in the theorem. We are given exact complexities by Theorem 4. In the second method, we apply Theorem 5 which allows us to combine two complexities previously computed. In the third method, we apply Theorem 6 which allows us to find low complexity normal elements from larger fields containing an optimal normal basis. The final method is based on a random search, which starts with a random finite field element, and randomly flips one of its polynomial coefficients a prescribed number of times, keeping the normal element found with lowest complexity. If Theorem 3 gives the lowest complexity then we indicate in the "Property" column of Table V which k value satisfies the conditions of the theorem. If Theorem 5 gives the lowest complexity, we indicate the coprime factorization of n that gives the low complexity. If the method of Theorem 6 achieves the lowest complexity, we note in the "Property" column which type of normal basis and which value of k is used. Theorem 5 does not apply for n prime or prime-power, and in the absence of an optimal basis for degree n or for a multiple of degree n, the random search is required. However, random search yields quite large smallest found complexities, and so appears only for few values of n. We observe that the smallest complexity found is not necessarily the minimum complexity possible in the field and for this reason the best found complexity is denoted as min c_N.

TABLE IV
STATISTICS ON THE COMPLEXITY OF NORMAL ELEMENTS OBTAINED USING THE `GrayCodeNCD` AND THE `SelfDualNCD` ALGORITHMS.

	normal element						**self-dual**		
n	$\frac{\Phi_2(x^n-1)}{n}$	m_{c_N}	M_{c_N}	Avg_{c_N}	Var_{c_N}	σ_{c_N}	m'_{c_N}	M'_{c_N}	Notes
2	1	3	3	3.00	0	0	3	3	Optimal, sd
3	1	5	5	5.00	0	0	5	5	Optimal, sd
4	2	7	9	8.00	1.00	1.00	-	-	
5	3	9	15	11.67	6.20	2.49	9	9	Optimal, sd
6	4	11	17	15.00	6.00	2.45	11	15	Optimal, sd
7	7	19	27	23.00	9.12	3.02	21	21	[10]: 3n-2
8	16	21	35	29.00	11.02	3.32	-	-	[10]: 3n-3
9	21	17	45	35.57	41.60	6.45	17	29	Optimal, sd
10	48	19	61	44.83	61.31	7.83	27	51	
11	93	21	71	55.82	57.61	7.59	21	57	Optimal, sd
12	128	23	83	64.13	139.48	11.81	-	-	
13	315	45	101	78.38	71.06	8.43	45	81	Best, sd
14	448	27	135	91.07	108.37	10.41	27	135	Optimal, sd
15	675	45	137	105.89	127.46	11.29	45	105	Best, sd
16	2048	85	157	115.82	731.70	27.05	-	-	
17	3825	81	177	132.77	671.84	25.92	81	171	Best, sd
18	5376	35	243	153.51	189.50	13.77	35	243	Optimal, sd
19	13797	117	229	172.00	174.05	13.19	117	201	Best, sd
20	24576	63	257	190.80	207.28	14.40	-	-	
21	27783	95	277	210.97	216.43	14.71	105	237	
22	95232	63	363	231.93	238.56	15.45	63	363	[10]: 3n-3
23	182183	45	325	254.02	254.60	15.96	45	309	Optimal, sd
24	262144	105	375	276.82	281.01	16.76	-	-	
25	629145	93	383	301.01	300.37	17.33	93	357	Best, sd
26	1290240	51	555	325.96	328.59	18.13	51	555	Optimal, sd
27	1835001	141	443	351.99	351.38	18.75	141	413	
28	3670016	55	517	378.98	379.12	19.47	-	-	Optimal
29	9256395	57	521	407.00	406.22	20.15	57	465	Optimal, sd
30	11059200	59	759	435.95	438.52	20.94	59	759	Optimal, sd
31	28629151	237	587	466.00	465.20	21.57	237	537	Best, sd
32	67108864	361	621	497.00	495.95	22.27	-	-	
33	97327197	65	693	529.00	528.48	22.99	65	693	Optimal, sd
34	250675200	243	819	562.00	561.52	23.70	243	819	Best, sd
35	352149515	69	779	596.00	595.03	24.39	69	693	Optimal, sd
36	704643060	71	1017	630.99	630.51	25.11	-	-	Optimal
37	1857283155	171	823	667.00	666.04	25.81			
38	3616800703	207	1131	704.00	703.18	26.52			
39	5282242828	77	933	742.00	741.09	27.22			Optimal

TABLE V
BEST FOUND COMPLEXITIES FOR $\mathbb{F}_{2^n}$ WITH $40 \leq n \leq 512$.

n	min c_N	Property	Method
40	189	5,8	Thm 5
41	81	Optimal	[18]
42	135	3,14	Thm 5
43	165	k = 4	Thm 3
44	147	4,11	Thm 5
45	153	5,9	Thm 5
46	135	2,23	Thm 5
47	261	k = 6	Thm 3
48	425	3,16	Thm 5
49	189	k = 4	Thm 3
50	99	Optimal	[18]
51	101	Optimal	[18]
52	103	Optimal	[18]
53	105	Optimal	[18]
54	209	Type 1, k = 3	Thm 6
55	189	5,11	Thm 5
56	399	7,8	Thm 5
57	585	3,19	Thm 5
58	115	Optimal	[18]
59	697	Type 2, k = 6	Thm 6
60	119	Optimal	[18]
61	345	k = 6	Thm 3
62	351	k = 6	Thm 3
63	323	7,9	Thm 5
64	1829	Prime Power	Random
65	129	Optimal	[18]
66	131	Optimal	[18]
67	261	k = 4	Thm 3
68	567	4,17	Thm 5
69	137	Optimal	[18]
70	207	2,35	Thm 5
71	841	Type 2, k = 6	Thm 6
72	357	8,9	Thm 5
73	285	k = 4	Thm 3
74	147	Optimal	[18]
75	465	3,25	Thm 5
76	297	k = 3	Thm 3
77	399	7,11	Thm 5
78	231	2,39	Thm 5
79	309	k = 4	Thm 3
80	765	5,16	Thm 5
81	161	Optimal	[18]
82	163	Optimal	[18]
83	165	Optimal	[18]
84	275	3,28	Thm 5
85	729	5,17	Thm 5
86	171	Optimal	[18]
87	285	3,29	Thm 5
88	441	8,11	Thm 5
89	177	Optimal	[18]
90	179	Optimal	[18]
91	525	k = 6	Thm 3
92	315	4,23	Thm 5
93	365	k = 4	Thm 3
94	369	k = 3	Thm 3

n	min c_N	Property	Method
95	189	Optimal	[18]
96	1805	3,32	Thm 5
97	381	k = 4	Thm 3
98	195	Optimal	[18]
99	197	Optimal	[18]
100	199	Optimal	[18]
101	585	k = 6	Thm 3
102	303	2,51	Thm 5
103	597	k = 6	Thm 3
104	945	8,13	Thm 5
105	209	Optimal	[18]
106	211	Optimal	[18]
107	621	k = 6	Thm 3
108	627	k = 5	Thm 3
109	1081	Type 2, k = 5	Thm 6
110	399	10,11	Thm 5
111	2201	Type 2, k = 10	Thm 6
112	1615	7,16	Thm 5
113	225	Optimal	[18]
114	663	k = 5	Thm 3
115	405	5,23	Thm 5
116	399	4,29	Thm 5
117	765	9,13	Thm 5
118	687	k = 6	Thm 3
119	237	Optimal	[18]
120	945	3,40	Thm 5
121	705	k = 6	Thm 3
122	711	k = 6	Thm 3
123	405	3,41	Thm 5
124	489	Type 1, k = 3	Thm 6
125	729	k = 6	Thm 3
126	459	9,14	Thm 5
127	501	k = 4	Thm 3
128	7821	Prime Power	Random
129	1281	Type 2, k = 5	Thm 6
130	259	Optimal	[18]
131	261	Optimal	[18]
132	455	4,33	Thm 5
133	2223	7,19	Thm 5
134	267	Optimal	[18]
135	269	Optimal	[18]
136	1701	8,17	Thm 5
137	801	k = 6	Thm 3
138	275	Optimal	[18]
139	549	k = 4	Thm 3
140	483	4,35	Thm 5
141	1681	Type 2, k = 6	Thm 6
142	831	k = 6	Thm 3
143	837	k = 6	Thm 3
144	1445	9,16	Thm 5
145	513	5,29	Thm 5
146	291	Optimal	[18]
147	861	k = 6	Thm 3
148	295	Optimal	[18]
149	2073	Type 2, k = 7	Thm 6
150	495	3,50	Thm 5
151	885	k = 6	Thm 3
152	2457	8,19	Thm 5
153	605	k = 4	Thm 3
154	567	11,14	Thm 5
155	309	Optimal	[18]
156	515	3,52	Thm 5
157	1561	Type 2, k = 5	Thm 6
158	315	Optimal	[18]
159	525	3,53	Thm 5
160	3249	5,32	Thm 5
161	855	7,23	Thm 5
162	323	Optimal	[18]
163	645	k = 4	Thm 3
164	567	4,41	Thm 5
165	585	5,33	Thm 5
166	495	2,83	Thm 5
167	2325	Type 2, k = 7	Thm 6
168	1995	3,56	Thm 5
169	669	k = 4	Thm 3
170	999	k = 6	Thm 3
171	1989	9,19	Thm 5

n	min c_N	Property	Method
172	343	Optimal	[18]
173	345	Optimal	[18]
174	347	Optimal	[18]
175	693	k = 4	Thm 3
176	1785	11,16	Thm 5
177	701	k = 4	Thm 3
178	355	Optimal	[18]
179	357	Optimal	[18]
180	359	Optimal	[18]
181	1065	k = 6	Thm 3
182	721	Type 1, k = 3	Thm 6
183	365	Optimal	[18]
184	945	8,23	Thm 5
185	2209	Type 2, k = 6	Thm 6
186	371	Optimal	[18]
187	1101	k = 6	Thm 3
188	1107	k = 5	Thm 3
189	377	Optimal	[18]
190	567	2,95	Thm 5
191	381	Optimal	[18]
192	9145	3,64	Thm 5
193	765	k = 4	Thm 3
194	387	Optimal	[18]
195	645	3,65	Thm 5
196	391	Optimal	[18]
197	3529	Type 2, k = 9	Thm 6
198	591	2,99	Thm 5
199	789	k = 4	Thm 3
200	1953	8,25	Thm 5
201	4001	Type 2, k = 10	Thm 6
202	1191	k = 6	Thm 3
203	1083	7,29	Thm 5
204	707	4,51	Thm 5
205	729	5,41	Thm 5
206	817	Type 1, k = 3	Thm 6
207	765	9,23	Thm 5
208	3825	13,16	Thm 5
209	417	Optimal	[18]
210	419	Optimal	[18]
211	2101	Type 2, k = 5	Thm 6
212	735	4,53	Thm 5
213	845	k = 4	Thm 3
214	849	k = 3	Thm 3
215	1269	k = 6	Thm 3
216	2961	8,27	Thm 5
217	1281	k = 6	Thm 3
218	1287	k = 5	Thm 3
219	869	k = 4	Thm 3
220	873	Type 1, k = 3	Thm 6
221	441	Optimal	[18]
222	735	3,74	Thm 5
223	2665	Type 2, k = 6	Thm 6
224	6859	7,32	Thm 5
225	1581	9,25	Thm 5
226	451	Optimal	[18]
227	5877	Type 2, k = 13	Thm 6
228	2255	Type 1, k = 9	Thm 6
229	5017	Type 2, k = 11	Thm 6
230	459	Optimal	[18]
231	461	Optimal	[18]
232	1197	8,29	Thm 5
233	465	Optimal	[18]
234	867	9,26	Thm 5
235	933	k = 4	Thm 3
236	937	Type 1, k = 3	Thm 6
237	2361	Type 2, k = 5	Thm 6
238	711	2,119	Thm 5
239	477	Optimal	[18]
240	3825	3,80	Thm 5
241	1425	k = 6	Thm 3
242	1431	k = 6	Thm 3
243	485	Optimal	[18]
244	969	k = 3	Thm 3
245	489	Optimal	[18]
246	815	3,82	Thm 5
247	1461	k = 6	Thm 3
248	4977	8,31	Thm 5

n	min c_N	Property	Method
249	825	3,83	Thm 5
250	5479	Type 2, k = 11	Thm 6
251	501	Optimal	[18]
252	935	9,28	Thm 5
253	945	11,23	Thm 5
254	507	Optimal	[18]
255	909	5,51	Thm 5
256	N/A	Prime Power	No data
257	1521	k = 6	Thm 3
258	855	3,86	Thm 5
259	2581	Type 2, k = 5	Thm 6
260	903	4,65	Thm 5
261	521	Optimal	[18]
262	783	2,131	Thm 5
263	1557	k = 6	Thm 3
264	1365	8,33	Thm 5
265	945	5,53	Thm 5
266	1575	k = 6	Thm 3
267	885	3,89	Thm 5
268	535	Optimal	[18]
269	3753	Type 2, k = 7	Thm 6
270	539	Optimal	[18]
271	1605	k = 6	Thm 3
272	6885	16,17	Thm 5
273	545	Optimal	[18]
274	2731	Type 2, k = 5	Thm 6
275	1953	11,25	Thm 5
276	959	4,69	Thm 5
277	1101	k = 4	Thm 3
278	555	Optimal	[18]
279	1109	k = 4	Thm 3
280	1449	8,35	Thm 5
281	561	Optimal	[18]
282	1671	k = 6	Thm 3
283	1677	k = 6	Thm 3
284	1129	Type 1, k = 3	Thm 6
285	945	3,95	Thm 5
286	1071	11,26	Thm 5
287	1539	7,41	Thm 5
288	6137	9,32	Thm 5
289	3457	Type 2, k = 6	Thm 6
290	1035	5,58	Thm 5
291	1725	k = 6	Thm 3
292	583	Optimal	[18]
293	585	Optimal	[18]
294	975	3,98	Thm 5
295	6469	Type 2, k = 11	Thm 6
296	10773	8,37	Thm 5
297	1761	k = 6	Thm 3
298	1767	k = 6	Thm 3
299	597	Optimal	[18]
300	995	3,100	Thm 5
301	3001	Type 2, k = 5	Thm 6
302	1201	Type 1, k = 3	Thm 6
303	605	Optimal	[18]
304	9945	16,19	Thm 5
305	1809	k = 6	Thm 3
306	611	Optimal	[18]
307	1221	k = 4	Thm 3
308	1155	11,28	Thm 5
309	617	Optimal	[18]
310	927	2,155	Thm 5
311	1845	k = 6	Thm 3
312	1617	8,39	Thm 5
313	1857	k = 6	Thm 3
314	1863	k = 5	Thm 3
315	1173	9,35	Thm 5
316	631	Optimal	[18]
317	8217	Type 2, k = 13	Thm 6
318	1055	3,106	Thm 5
319	1197	11,29	Thm 5
320	16461	5,64	Thm 5
321	3841	Type 2, k = 6	Thm 6
322	1215	14,23	Thm 5
323	645	Optimal	[18]
324	1127	4,81	Thm 5
325	1293	k = 4	Thm 3

n	min c_N	Property	Method
326	651	Optimal	[18]
327	14345	Type 2, k = 22	Thm 6
328	1701	8,41	Thm 5
329	657	Optimal	[18]
330	659	Optimal	[18]
331	1965	k = 6	Thm 3
332	1155	4,83	Thm 5
333	8721	9,37	Thm 5
334	2629	k = 7	Thm 3
335	4009	Type 2, k = 6	Thm 6
336	8075	3,112	Thm 5
337	3361	Type 2, k = 5	Thm 6
338	675	Optimal	[18]
339	1125	3,113	Thm 5
340	1353	k = 3	Thm 3
341	4081	Type 2, k = 6	Thm 6
342	2031	k = 6	Thm 3
343	1365	k = 4	Thm 3
344	15771	8,43	Thm 5
345	1233	5,69	Thm 5
346	691	Optimal	[18]
347	2061	k = 6	Thm 3
348	695	Optimal	[18]
349	3481	Type 2, k = 5	Thm 6
350	699	Optimal	[18]
351	3501	Type 2, k = 5	Thm 6
352	7581	11,32	Thm 5
353	4929	Type 2, k = 7	Thm 6
354	707	Optimal	[18]
355	2109	k = 6	Thm 3
356	1239	4,89	Thm 5
357	1185	3,119	Thm 5
358	1071	2,179	Thm 5
359	717	Optimal	[18]
360	3213	5,72	Thm 5
361	10801	Type 2, k = 15	Thm 6
362	2151	k = 5	Thm 3
363	1445	k = 4	Thm 3
364	1449	k = 3	Thm 3
365	9465	Type 2, k = 13	Thm 6
366	1095	2,183	Thm 5
367	2181	k = 6	Thm 3
368	3825	16,23	Thm 5
369	1377	9,41	Thm 5
370	1323	5,74	Thm 5
371	741	Optimal	[18]
372	743	Optimal	[18]
373	1485	k = 4	Thm 3
374	1489	Type 1, k = 3	Thm 6
375	749	Optimal	[18]
376	19173	8,47	Thm 5
377	2565	13,29	Thm 5
378	755	Optimal	[18]
379	6805	Type 2, k = 9	Thm 6
380	1323	4,95	Thm 5
381	7601	Type 2, k = 10	Thm 6
382	1143	2,191	Thm 5
383	16045	Type 2, k = 21	Thm 6
384	39105	3,128	Thm 5
385	1449	11,35	Thm 5
386	771	Optimal	[18]
387	1541	k = 4	Thm 3
388	775	Optimal	[18]
389	14745	Type 2, k = 19	Thm 6
390	1295	3,130	Thm 5
391	2325	k = 6	Thm 3
392	21021	8,49	Thm 5
393	785	Optimal	[18]
394	3915	Type 1, k = 9	Thm 6
395	2349	k = 6	Thm 3
396	1379	4,99	Thm 5
397	2361	k = 6	Thm 3
398	795	Optimal	[18]
399	4777	Type 2, k = 6	Thm 6
400	7905	16,25	Thm 5
401	4801	Type 2, k = 6	Thm 6
402	1335	3,134	Thm 5

n	min c_N	Property	Method
403	8845	Type 2, k = 11	Thm 6
404	1609	Type 1, k = 3	Thm 6
405	1449	5,81	Thm 5
406	1539	14,29	Thm 5
407	10773	11,37	Thm 5
408	2121	8,51	Thm 5
409	1629	k = 4	Thm 3
410	819	Optimal	[18]
411	821	Optimal	[18]
412	1641	Type 1, k = 3	Thm 6
413	825	Optimal	[18]
414	827	Optimal	[18]
415	1485	5,83	Thm 5
416	16245	13,32	Thm 5
417	1661	k = 4	Thm 3
418	835	Optimal	[18]
419	837	Optimal	[18]
420	839	Optimal	[18]
421	11761	Type 2, k = 14	Thm 6
422	10947	Type 2, k = 13	Thm 6
423	1685	k = 4	Thm 3
424	2205	8,53	Thm 5
425	2529	k = 6	Thm 3
426	851	Optimal	[18]
427	7669	Type 2, k = 9	Thm 6
428	2547	k = 5	Thm 3
429	857	Optimal	[18]
430	1539	5,86	Thm 5
431	861	Optimal	[18]
432	11985	16,27	Thm 5
433	1725	k = 4	Thm 3
434	4315	Type 1, k = 9	Thm 6
435	1733	k = 4	Thm 3
436	14727	Type 1, k = 33	Thm 6
437	5265	19,23	Thm 5
438	875	Optimal	[18]
439	4381	Type 2, k = 5	Thm 6
440	3969	5,88	Thm 5
441	881	Optimal	[18]
442	883	Optimal	[18]
443	885	Optimal	[18]
444	1475	3,148	Thm 5
445	1593	5,89	Thm 5
446	2655	k = 6	Thm 3
447	2661	k = 6	Thm 3
448	34751	7,64	Thm 5
449	6273	Type 2, k = 7	Thm 6
450	1683	9,50	Thm 5
451	1701	11,41	Thm 5
452	1575	4,113	Thm 5
453	905	Optimal	[18]
454	9025	Type 1, k = 19	Thm 6
455	2451	7,65	Thm 5
456	12285	3,152	Thm 5
457	13681	Type 2, k = 15	Thm 6
458	2727	k = 6	Thm 3
459	4581	Type 2, k = 5	Thm 6
460	919	Optimal	[18]
461	2745	k = 6	Thm 3
462	1383	2,231	Thm 5
463	5545	Type 2, k = 6	Thm 6
464	4845	16,29	Thm 5
465	1545	3,155	Thm 5
466	931	Optimal	[18]
467	2781	k = 6	Thm 3
468	1751	9,52	Thm 5
469	1869	k = 4	Thm 3
470	939	Optimal	[18]
471	9401	Type 2, k = 10	Thm 6
472	32067	8,59	Thm 5
473	945	Optimal	[18]
474	1575	3,158	Thm 5
475	1893	k = 4	Thm 3
476	1659	4,119	Thm 5
477	1785	9,53	Thm 5
478	1431	2,239	Thm 5
479	5737	Type 2, k = 6	Thm 6
480	16245	3,160	Thm 5
481	2865	k = 6	Thm 3
482	2871	k = 5	Thm 3
483	965	Optimal	[18]
484	1929	Type 1, k = 3	Thm 6
485	8713	Type 2, k = 9	Thm 6
486	1455	2,243	Thm 5
487	1941	k = 4	Thm 3
488	34041	8,61	Thm 5
489	5857	Type 2, k = 6	Thm 6
490	979	Optimal	[18]
491	981	Optimal	[18]
492	1863	12,41	Thm 5
493	1965	k = 4	Thm 3
494	1969	Type 1, k = 3	Thm 6
495	989	Optimal	[18]
496	20145	16,31	Thm 5
497	9921	Type 2, k = 10	Thm 6
498	1815	6,83	Thm 5
499	1989	k = 4	Thm 3
500	5969	Type 1, k = 11	Thm 6
501	5001	Type 2, k = 5	Thm 6
502	1503	2,251	Thm 5
503	2997	k = 6	Thm 3
504	6783	7,72	Thm 5
505	5041	Type 2, k = 5	Thm 6
506	2835	2,253	Thm 5
507	2021	k = 4	Thm 3
508	1015	Optimal	[18]
509	1017	Optimal	[18]
510	1919	10,51	Thm 5
511	3045	k = 6	Thm 3
512	N/A	Prime Power	No data

C. Conjectures

To examine the distribution of normal bases of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, in Fig. 3 we plot the number of normal bases found against their complexity. In each case, we observe a Gaussian shaped curve. This leads to the following conjecture.

Conjecture 1: The number of normal bases for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ is normally distributed with respect to their complexities.

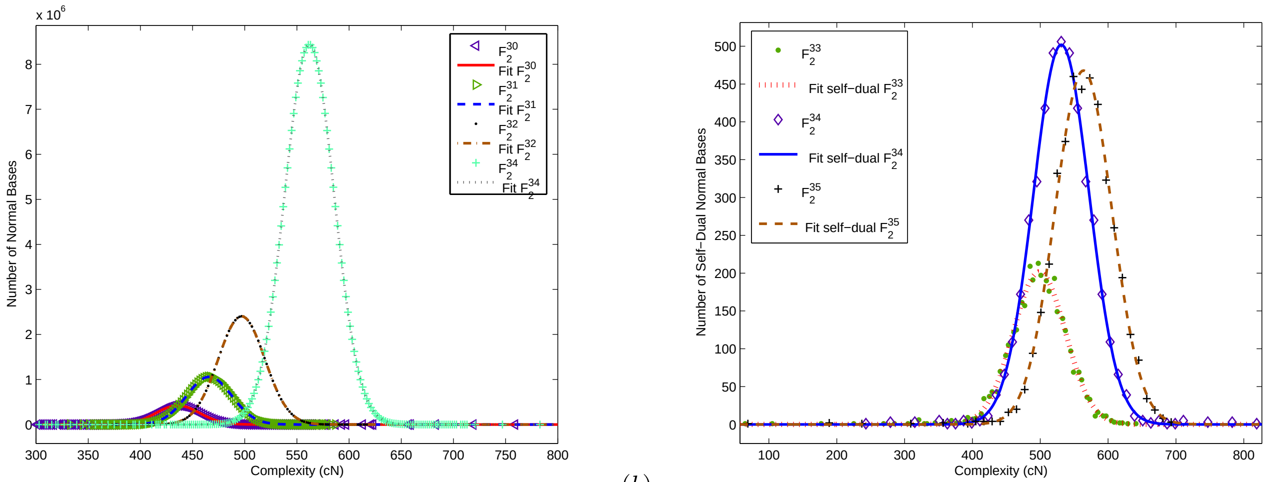
This conjecture is verified for $n \leq 39$ by running the Shapiro-Wilk normality test [19] on the results of the `GrayCodeNCD` experiments and, for $15 \leq n \leq 39$, is guaranteed with over 99.9% certainty. As a consequence of this conjecture, although probabilistic algorithms to find normal elements exist [12], these will not give low complexity normal elements. For efficient computation, new searching methods for normal elements must be developed.

Finding the general form of the distribution curve requires calculating the average complexity and the variance. Alternatively, an enveloping curve requires bounds (as functions of n) to find the averages and variances of the complexities. This is the motivation for the Conjectures 2 and 3, which suggest upper bounds on the average and variance of the complexities of normal bases.

The data lends itself to the following conjecture on a bound for the average complexity of normal elements.

Conjecture 2: The average complexity of normal elements in a finite field $\mathbb{F}_{q^n}$ with $n \geq 8$ has an upper bound of $(n^2 - n + 3)/2$.

This conjecture is based upon the data found using the `GrayCodeNCD` algorithm, and observing the symmetry of the Gaussian curve conjectured previously. This is very close to half of the largest possible complexity $n^2 - n$, which is



(a) (b)

Fig. 3. Sample distributions of complexity of (a) normal elements and (b) self-dual normal elements.

always even, so the extra factor of adding 3 in the numerator accounts for any decimal places calculated in the average. This conjectured bound is quite tight for many values of $n \leq 39$, and only for $n = 14, 23, 25$ the numerator requires a constant term of 3 rather than 2.

Conjecture 3: For sufficiently large n the variance of complexities of normal bases is bounded above by $n^2/2$.

This conjecture comes from the side-by-side comparison in Table IV of the average and variance of the complexities. For this data $n \geq 19$ suffices. The variance and the average complexity are quite similar, which implies that there could probably be a stronger conjecture here, since this conjecture is loose in comparison to the one on the average complexity.

We recall that the average complexity of self-dual normal elements is given in Theorem 9, so the next step would naturally be to conjecture on the variance of the complexity of self-dual normal elements. However, since there are very few self-dual normal bases for $n \leq 36$ more experiments are required to give sufficient data to support any such conjecture.

The following argument leads to our final conjecture. Let us assume that the above conjectures on the average and variance of complexities of all normal elements in a finite field (i.e, not restricted to self-dual normal elements) hold. Moreover, let us consider the following problem: is there some constant k such that the minimum complexity element in a finite field $\mathbb{F}_{2^n}$ is bounded above by kn? We observe that the probability of finding a normal element of complexity $c_N \leq kn$ is analogous to finding the density under the normalized curve as follows, $P(c_N \leq kn) = P[Z \leq (kn - \mu)/\sigma]$. A low complexity is analogous to a low Z-score on the normalized curve. Given that, by our conjectures, $\mu = (n^2 - n + 3)/2$ and $\sigma^2 = n^2/2$, calculating the Z-score of kn gives $\frac{kn - \frac{n^2-n+3}{2}}{n/\sqrt{2}}$. We observe now that if k is a constant, then the Z-score becomes infinitely small. Relating this to the complexity distribution, this implies that the upper bound on the minimum complexity vanishes, which is a contradiction since the minimum possible complexity is $2n - 1$.

Conjecture 4: There is no constant k such that the complexity C_n of $\mathbb{F}_{2^n}$ is bounded above by kn for all n. Furthermore, if the average and variance of the complexities of all normal elements in $\mathbb{F}_{2^n}$ are both of order n^2, then C_n is also of order n^2.

Our final observation concerns the distribution of the normal elements themselves. To achieve results for $36 \leq n \leq 39$ we distributed the computation across many processors by letting each processor deal with a range of field elements that were consecutive in the Gray code order. This divided our time linearly with the number of processors used. For example, the estimated run time of $n = 39$ was 8 months using a single processor, but we were able to complete the simulation in just under 3 weeks using 13 processors. Furthermore, we observed that there were large blocks along the run of the Gray code in which there were no lexicographically canonical normal elements found. This is an interesting topic for future research about the existence of normal elements with prescribed coefficients.

Acknowledgment

We would like to thank the two anonymous reviewers for their careful review of this paper, and for all of their insightful comments and suggestions.

References

[1] D. W. Ash, I. F. Blake, and S. A. Vanstone, Low complexity normal bases, *Discrete Applied Mathematics*, **25** (1989), 191-210.

[2] T. Beth, W. Geiselmann, and F. Meyer, Finding (good) normal bases in finite fields, *Proceedings of the International Conference on Symbolic and Algebraic Computation*, (1991), 173-178.

[3] I. F. Blake, S. Gao, and R. C. Mullin, Explicit factorization of $x^{2^k} + 1$ over $\mathbb{F}_p$ with prime $p \equiv 3 \bmod 4$, *Applicable Algebra in Engineering, Communication and Computing*, **4** (1993), 89-94.

[4] M. Christopoulou, T. Garefalakis, D. Panario, and D. Thomson, The trace of an optimal normal element and low complexity normal bases, *Designs, Codes and Cryptography, to appear*, 2007.

[5] R. M. Avanzi, H. Cohen, C. Douche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren (ed.), *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Discrete Mathematics and its Applications Series, Chapman and Hall/CRC, 2006.

[6] G. S. Frandsen, On the density of normal bases in finite fields, *Finite Fields and Their Applications*, **6** (2000), 23-38.

[7] S. Gao, J. von zur Gathen, D. Panario, and V. Shoup, Algorithms for exponentiation in finite fields, *Journal of Symbolic Computation*, **29** (2000), 879-889.
[8] S. Gao and H. W. Lenstra, Optimal normal bases, *Designs, Codes and Cryptography*, **2** (1992), 315-323.
[9] S. Gao and D. Panario, Density of normal elements, *Finite Fields and Their Applications*, **3** (1997), 141-150.
[10] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, 2nd edition, 2003.
[11] J. von zur Gathen and J. Gerhard, Polynomial factorization over $\mathbb{F}_2$, *Mathematics of Computation*, **71** (2002), 1677-1698.
[12] J. von zur Gathen and M. Giesbrecht, Constructing normal bases in finite fields, *Journal of Symbolic Computation*, **10** (1990), 547-570.
[13] W. Geiselmann, *Algebraische Algorithmenentwicklung am Beispiel der Arithmetik in endlichen Körpern*, Dissertation, Universität Karlsruhe, 1992.
[14] K. Hensel, Über die Darstellung der Zahlen eines Gattungsbereiches für einen beliebigen Primdivisor, *Journal für die reine und angewandte Mathematik*, **103** (1888), 230-237.
[15] D. Jungnickel, *Finite Fields: Structure and Arithmetics*, B.I. Wissenschaftsverlag, Mannheim, 1993.
[16] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, 2nd edition, 1994.
[17] F. Meyer, Normalbasismultiplikation in endlichen Körpern; Diplomarbeit, University of Karlsruhe, 1990.
[18] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson, Optimal normal bases in $GF(p^n)$, *Discrete Applied Mathematics*, **22** (1988/1989), 149-161.
[19] M. Mendez and A. Pala, Type I error rate and power of three normality tests, *Pakistan Journal of Information and Technology*, **2** (2003), 135-139.
[20] P. Ning and Y. Yin, Efficient software implementation for finite field multiplication in normal basis, *Lecture Notes in Computer Science*, **2229** (2001), 177-188.
[21] C. Savage, A survey of combinatorial Gray codes, *SIAM Review*, **39** (1997), 605-629.
[22] V. Shoup, Number Theory Library (NTL), Version 5.4, `http://www.shoup.net`, 2006.
[23] D. Stinson, Some observations on parallel algorithms for fast exponentiation in $\mathbb{F}_{2^n}$, *SIAM Journal on Computing*, **19** (1990), 711-717.
[24] C. C. Wang, An algorithm to design finite field multipliers using a self-dual normal basis, *IEEE Transactions on Computers*, **38** (1989), 1457-1460.
[25] Z.-X. Wan and K. Zhou, On the complexity of the dual basis of a type I optimal normal basis, *Finite Fields and Their Applications*, **13** (2007), 411-417.
[26] B. Young and D. Panario, Low complexity normal bases in $\mathbb{F}_{2^n}$, *Finite Fields and Their Applications*, **10** (2004), 53-64.

Ariane M. Masuda received her Bachelor's degree in Mathematics from the Federal University of Paraná, her Master's degree in Mathematics from the Fluminense Federal University, and her Ph.D. in Mathematics from Carleton University. She is currently an NSERC postdoctoral fellow at the University of Ottawa. Her research interests include algebra, number theory, cryptography, and coding theory.

Lucia Moura was born in Brazil, where she completed her Bachelor's and Master's degrees in Computer Science, at the University of São Paulo. She completed her Ph.D. in Computer Science at the University of Toronto in 1999, and joined the School of Information Technology and Engineering of the University of Ottawa (Canada) in 2000, where she is currently an Associate Professor. Her main research areas are algorithms and combinatorics, with special interest in combinatorial algorithms, combinatorial designs and their applications.

Daniel Panario was born in Uruguay where he studied Mathematics and Computer Science. He received a M.Sc. degree from the University of São Paulo (Brazil) and a Ph.D. from the University of Toronto (Canada). He is an Associate Professor at Carleton University in Ottawa (Canada). His main research interests are in finite fields and applications, and in analysis of algorithms.

David Thomson was raised in the Ottawa, Canada area and completed his B.Math in 2006 and M.Sc. in Mathematics in 2007 from Carleton University (Canada). He is currently a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Waterloo (Canada). His research interests include finite fields and their applications, and in cryptography and security.