Some results on arithmetic codes of composite length

Tai-Yang Hwang
Syracuse University

Carlos R.P. Hartmann
Syracuse University, chartman@syr.edu

Follow this and additional works at: https://surface.syr.edu/eecs_techreports

Part of the Computer Sciences Commons

Recommended Citation
https://surface.syr.edu/eecs_techreports/6

This Report is brought to you for free and open access by the College of Engineering and Computer Science at SURFACE. It has been accepted for inclusion in Electrical Engineering and Computer Science - Technical Reports by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.
SOME RESULTS ON ARITHMETIC CODES OF COMPOSITE LENGTH

Tai-Yang Hwang
Carlos R. P. Hartmann

October 1976
SOME RESULTS ON ARITHMETIC CODES OF
COMPOSITE LENGTH

Tai-Yang Hwang
Carlos R. P. Hartmann

School of Computer and Information Science
Syracuse University
Syracuse, New York 13210
Tel. (315) 423-2368

This work was supported by the National Science Foundation under
Grant ENG75-07709.
ABSTRACT

In this paper we present a new upper bound on the minimum distance of binary cyclic arithmetic codes of composite length. Two new classes of binary cyclic arithmetic codes of composite length are introduced. The error correction capability of these codes are discussed and in some cases the actual minimum distance is found. Decoding algorithms based on majority-logic decision are proposed for these codes.
I. Introduction

Arithmetic codes, first proposed by Diamond [1], are useful for error control in digital computation as well as in data transmission. They are particularly suitable for checking or correcting errors in arithmetic processors. Finding the minimum distance $d$ of an arithmetic code is a major problem. Despite similarities between cyclic arithmetic and cyclic block codes, no general lower bound and, similar to the BCH bound for cyclic codes, exists for arithmetic codes. Thus, in general, the determination of $d$ still relies on computer search. The search for a systematic way of constructing arithmetic codes is another major area of research. Three known classes of arithmetic codes are the high-rate perfect single-error correcting codes [2]-[4], the large-distance low-rate Mandelbaum-Barrows codes [5],[6] and the intermediate-rate intermediate-distance codes [7]. One of the interesting features of the codes introduced in [7] is that they can be decoded using majority-logic decisions.

In this paper we present a new upper bound on $d$ for binary cyclic arithmetic codes of composite length. This bound is quite tight and gives a rather good estimation of the actual minimum distance. We also construct two new classes of binary cyclic arithmetic codes. Many of these codes have intermediate-rate and intermediate-distance and they can be decoded by majority-logic decisions.

In Section II, we present the new upper bound on $d$. In
Section III, we construct the two new classes of binary cyclic
arithmetic codes. The decoding algorithm for these codes are
given in Section IV. A discussion of the results is contained in
Section V. Numerical examples are given in Appendix A. The
conditions for the existence of codes in the classes constructed
in Section III are given in Appendix B.
II. Bound on the Minimum Distance of Binary Cyclic Arithmetic Codes of Composite Length

A binary cyclic arithmetic (AN) code of length \( n \) is of the form \( \text{AN} \), where \( A \) is a fixed integer, called the generator of the code, and \( N = 0,1, \ldots, B-1 \). \( B \) is chosen so that \( AB = 2^n-1 \), where \( n \) is the multiplicative order of 2 modulo \( A \). For a general background on binary cyclic AN-code as well as for the definitions of arithmetic distance and arithmetic weight, the readers are referred to [8]-[10].

The following theorem, which is a generalization of [11, Theorem 1], gives an upper bound on \( d \).

**Theorem 1:** Let \( \text{AN} \) be a binary arithmetic code of composite length \( n = \sum_{l=1}^{\ell_1} a_l l_1, \ 1 < \ell_1 < n \). If \( B \) is divisible by either \( 2^{\ell_1} + 1 \) or \( 2^{\ell_1} - 1 \), then \( d \leq \ell_1 \).

**Proof:** Let \( B = B_1(2^{\ell_1}+1) \). By [12, Lemma 6.3] \( \ell_1 \) is even. Thus,

\[
AB_1 = \frac{2^n-1}{2^{\ell_1}+1} = 2^{\ell_1} - 2^{\ell_1-2} + \ldots + 2^{\ell_1-1} - 1
\]

is a codeword of arithmetic weight \( \ell_1 \), \( W(AB_1) = \ell_1 \). Similarly, we can show that \( d \leq \ell_1 \) when \( B = B_2(2^{\ell_1}-1) \).

Q.E.D.

The following example will illustrate the application of Theorem 1.

**Example 1:** Let \( AB = 2^{20} - 1 \) with \( A = 5 \cdot 31 \cdot 41 \). Thus, \( B = 3 \cdot 5 \cdot 11 \) and \( n = 20 \). We note that \( \text{GCD}(A,2^2-1) = 1 \), \( \text{GCD}(A,2^4-1) \neq 1 \) and
GCD(A, 2^5 - 1) \neq 1. Thus, by [11, Theorem 1] d \leq 10. We may write

B = 5(2^5 + 1). Thus, by Theorem 1 d \leq 4. This code has d = 4 [13].

When B is of the form 2^n + 1 or 2^n - 1, the exact minimum distance
can be determined. This result is given in the following:

**Theorem 2**: When B = 2^n + 1 (or 2^n - 1), then d = n/n_1 = \frac{4}{5}.

**Proof**: For B = 2^n + 1,

\[
A = 2^{n_1-1} = 2^{(\frac{4}{5} - 1)n_1} - 2^{(\frac{4}{5} - 2)n_1} + \ldots + 2^{n - 1 - 1}
\]

\[
= (2^{1 - 1})(2^{(\frac{4}{5} - 2)n_1} + 2^{(\frac{4}{5} - 4)n_1} + \ldots + 2^{n_1} + 1).
\]

It is easily seen that W(AN) = \frac{4}{5} for N = 1, N = 2^n_1 and

N = 2^n - 1. If 0 < N - 1 < 2^n - 2, then

N - 1 = a_{n_1 - 1} 2^{n_1 - 1} + a_{n_1 - 2} 2^{n_1 - 2} + \ldots + a_1 2 + a_0

where a_i = 0 or 1, for i = 0, 1, ..., n_1 - 1. Furthermore not all a_i are 0 or 1. Thus,

\[
A(N - 1) = (2^{(\frac{4}{5} - 1)n_1} - 2^{(\frac{4}{5} - 2)n_1} + \ldots + 2^{n_1 - 1})(a_{n_1 - 1} 2^{n_1 - 2} + a_{n_1 - 2} 2^{n_1 - 2} + \ldots + a_1 2 + a_0)
\]

and

\[
AN = (a_{n_1 - 1} 2^{n_1 - 1} + a_{n_1 - 2} 2^{n_1 - 2} + \ldots + a_1 2 + a_0) 2^{(\frac{4}{5} - 1)n_1}
\]

\[
+ ((1 - a_{n_1 - 1}) 2^{n_1 - 1} + (1 - a_{n_1 - 2}) 2^{n_1 - 2} + \ldots + (1 - a_1) 2 + (1 - a_0)) 2^{(\frac{4}{5} - 2)n_1}
\]

\[
+ (a_{n_1 - 1} 2^{n_1 - 1} + a_{n_1 - 2} 2^{n_1 - 2} + \ldots + a_1 2 + a_0) 2^{(\frac{4}{5} - 3)n_1}
\]

\[
+ ((1 - a_{n_1 - 1}) 2^{n_1 - 1} + (1 - a_{n_1 - 2}) 2^{n_1 - 2} + \ldots + (1 - a_1) 2 + (1 - a_0)) 2^{(\frac{4}{5} - 4)n_1}
\]
By [7, Lemma 2], \( W(AN) \geq 1 \). Thus, \( d = 2 \) when \( B = 2^{n_1+1} \).

Erosh and Erosh [14] showed that \( d = 2 \) when \( B = 2^{n_1-1} \).

Q.E.D.

Example 2: Let \( AB = 2^8-1 \) with \( A = 3\cdot17 \). Thus \( B = 5 \) and \( n = 8 \).

By Theorem 2, \( d = 4 \).

Tables I and II in Appendix A give numerical examples of the application of Theorem 1 and 2, respectively.
III. On the Minimum Distance of Two Classes of Cyclic Arithmetic Codes of Composite Length

In this section we will consider two classes of cyclic AN-codes. The first class, $C_1$, has a generator of the form

$$A = \frac{2^{n-1}}{n_1(2^{n_1}+1)(2^{n_2}+1)}$$

and the second class, $C_2$, of the form

$$A = \frac{2^{n-1}}{(n_1+1)(n_2-1)}$$

where $n_1 \neq n_2$, $n = \ell_1 n_1 = \ell_2 n_2$ where $1 < \ell_1 < n$ and $1 < \ell_2 < n$. Appendix B gives the conditions for the existence of codes in these classes.

We first consider the class $C_1$. By [12, Lemma 6.3], $\ell_1$ and $\ell_2$ are even integers.

Theorem 3: If $n_2 > n_1$ then $d$ of the codes in $C_1$ is bounded by

$$\min(\ell_2, \ell_1/2) \leq d \leq \ell_2.$$  

Proof: By Theorem 1, $d \leq \ell_2$. To obtain the lower bound we proceed as follows: if $N \equiv 0 \mod (2^{n_1}+1)$, then $AN$ is a nonzero code-word in the AN-code generated by $(2^n-1)/(2^{n_1}+1)$ and by Theorem 2, $W(AN) \geq \ell_2$; if $N / 0 \mod (2^{n_1}+1)$, then $AN(2^{n_1}+1)$ is a nonzero codeword in the AN-code generated by $(2^n-1)/(2^{n_1}+1)$ and by Theorem 2, $W[AN(2^{n_1}+1)] \geq \ell_1$. By the triangle inequality we have

$$W[AN(2^{n_2}+1)] \leq W(AN \cdot 2^{n_2}) + W(AN),$$
which implies

\[ W[\Lambda N(2^{n_2+1})] \leq 2W(\Lambda N) \ . \]

Thus \( W(\Lambda N) \geq \ell_1/2 \).

Q.E.D.

**Example 3:** Let \( AB = 2^{60}-1 \) with \( B = (2^{15}+1)(2^{10}+1) \). Thus,
\[ A = (2^{60}-1)/(2^{15}+1)(2^{10}+1) \] and \( n = 60 \). By Theorem 3, \( 3 \leq d \leq 4 \).

Next, we consider the class \( C_2 \). By [12, Lemma 6.3], \( \ell_1 \) is even.

**Theorem 4:** For codes in the class \( C_2 \) the following hold:

(a) If \( n_2 > n_1 \), then \( \min(\ell_2, \ell_1/2) \leq d \leq \ell_2 \).

(b) If \( n_2 < n_1 \), then \( d = \ell_1 \).

Proof: If \( n_2 > n_1 \), then the proof is analogous to the proof of Theorem 3. If \( n_2 < n_1 \), then the proof is analogous to the proof of [7, Theorem 1].

Q.E.D.

**Example 4:** Let \( AB = 2^{60}-1 \) with \( B = (2^{15}+1)(2^{10}-1) \). Thus, 
\[ A = (2^{60}-1)/(2^{15}+1)(2^{10}-1) \] and \( n = 60 \). By Part (a) of Theorem 4, \( 3 \leq d \leq 4 \).

**Example 5:** Let \( AB = 2^{72}-1 \) with \( B = (2^{12}+1)(2^{9}-1) \). Thus, 
\[ A = (2^{72}-1)/(2^{12}+1)(2^{9}-1) \] and \( n = 72 \). By Part (b) of Theorem 4, \( d = 6 \).

Tables III and IV in Appendix A give numerical examples of the application of Theorems 3 and 4.
IV. Decoding Class \( C_1 \) and Class \( C_2 \) Codes

In this section we will present decoding algorithms for the codes of Classes \( C_1 \) and \( C_2 \). Their decoding algorithms depend on the decoding of the codes of length \( n = n_1 \ell_1 \) generated by 
\[
A_0 = \frac{2^n - 1}{2^{\ell_1} + 1},
\]
which by Theorem 2 has minimum distance \( \ell_1 \).

Suppose \( R = A_0 N + E, 0 \leq N \leq 2^{n_1} \), is a corrupted codeword, and the arithmetic weight of the error pattern is \( W(E) = t \leq \lfloor (\ell_1 - 1)/2 \rfloor \), where \( \lfloor x \rfloor \) denotes the largest integer less than or equal to \( x \).

As the first step of decoding we note that \( N \) is equal to zero if and only if \( W(R) = W(E) \leq \lfloor (\ell_1 - 1)/2 \rfloor \). Thus, \( N = 0 \) can be uniquely identified. When \( 0 < N \leq 2^{n_1} \), the decoding will be based on the result of the following theorem:

Theorem 5: The binary form of a codeword \( A_0 N, 0 < N < 2^{n_1} \), is the following:

\[
A_0 N = (a_{n_1-1})^2 + a_{n_1-2}^2 + \ldots + a_1^2 + a_0^2
\]

\[
+ ((1-a_{n_1-1})^2 + (1-a_{n_1-2})^2 + \ldots + (1-a_1)^2 + (1-a_0))^2
\]

\[
+ (a_{n_1-1})^2 + a_{n_1-2}^2 + \ldots + a_1^2 + a_0^2
\]

\[
+ ((1-a_{n_1-1})^2 + (1-a_{n_1-2})^2 + \ldots + (1-a_1)^2 + (1-a_0))^2
\]

\[
+ \ldots
\]

\[
+ (a_{n_1-1})^2 + a_{n_1-2}^2 + \ldots + a_1^2 + a_0^2
\]

\[
+ ((1-a_{n_1-1})^2 + (1-a_{n_1-2})^2 + \ldots + (1-a_1)^2 + (1-a_0))^2
\]
where $a_i$ is 0 or 1 for $i = 0, 1, \ldots, n_1 - 1$ and

$$N - 1 = a_{n_1 - 1}^{n_1 - 1} + a_{n_1 - 2}^{n_1 - 2} + \ldots + a_1 + a_0.$$  

The proof of Theorem 5 is similar to the proof of Theorem 2.

Since the carry propagation caused by an error stops whenever a digit 0 is reached and the borrow propagation caused by an error stops whenever a digit 1 is reached, then, by Theorem 5, a single error can never corrupt more than $n_1 - 1$ consecutive digits in the binary form of $A_0N$ when $N \neq 2^i$, $i = 0, 1, \ldots, n_1$; and a single error can never corrupt more than $n_1 + 1$ consecutive digits in the binary form of $A_0N$ when $N = 2^i$, $i = 0, 1, \ldots, n_1$.

Let $A_0N = \sum_{j=0}^{n-1} b_j 2^j$, $b_j$ is 0 or 1 for $j = 0, 1, \ldots, n-1$.

If a single error does not corrupt more than $n_1$ digits, the binary coefficient $b_k$, $0 \leq k < n_1$, can be correctly estimated by taking the majority vote on the coefficients $b_k$, $1 - b_{k+n_1}$, $b_{k+2n_1}$, $1 - b_{k+3n_1}$, $\ldots$, $b_{k+(l_1-2)n_1}$ and $1 - b_{k+(l_1-1)n_1}$ whenever $W(E) \leq (l_1 - 2)/2$ [7]. Thus, if $N \neq 2^i$, $i = 0, 1, \ldots, n_1$ and $W(E) \leq (l_1 - 2)/2$ we would, using the above majority decision, correctly estimate $A_0N$. If $N = 2^i$, $i = 0, 1, \ldots, n_1$, a single error can corrupt $n_1 + 1$ consecutive digits, this can contribute to at most two wrong votes in the majority decision. However, by noting the following facts:

(a) a carry propagation caused by an error which corrupts $n_1 + 1$ digits will introduce a subsequence

of the form $F_1 = (10 \ldots 0)$ with at least $n_1 + 1$

consecutive 0's; and
(b) a borrow propagation caused by an error which corrupts \( n_1 + 1 \) digits, will introduce a subsequence of the form \( F_2 = (01 \ldots 1) \) with at least \( n_1 + 1 \) consecutive 1's, we can remove the effect of \( n_1 + 1 \) corrupted consecutive digits by applying the following operation on the binary representation of \( R 

\text{Operation 1:} \quad \text{If there is a subsequence of the form } F_1, \text{ then change it to } F_1' = (10 \ldots 010 \ldots 0) \text{ by changing the } (n_1 + 1)\text{th bit of } F_1 \text{ from 0 to 1. If there is a subsequence of the form } F_2, \text{ then change it to } F_2' = (01 \ldots 101 \ldots 1) \text{ by changing the } (n_1 + 1)\text{th bit of } F_2 \text{ from 1 to 0.}

Thus, if \( N = 2^i \), in the modified binary representation of \( R \), each error will contribute to at most one wrong vote in the majority decision.

If \( N \neq 2^i \), for \( i = 0, 1, \ldots, n_1 \), Operation 1 will not change the majority decision since in this case it needs at least two errors to introduce a subsequence of the form \( F_1 \) or \( F_2 \).

In summary, the decoding of the AON code can be described as follows:

(a) If \( W(R) \leq \lceil \vartheta_1 - 1 \rceil / 2 \), decode \( N = 0 \), otherwise go to (b)

(b) If form \( F_1 \) or \( F_2 \) appears in the binary representation of \( R \) apply Operation 1; otherwise go to (c)

(c) the binary coefficients of \( N \) are determined by majority-logic decisions.

The decoding scheme described above for the AON codes can be used in the decoding of codes of Class C_1 and C_2. Let \( \text{res}(x) \) denote
the residue of $x$ modulo $2^{n_1}$. Let $R = AN + E$ be the received word while $AN$ is the codeword sent.

**Decoding algorithm for Class $C_1$ Codes** ($n_2 \geq 2n_1$)

(a) $N = 0$ if and only if $W(R) \leq \lfloor (\ell_1 - 1)/2 \rfloor$; otherwise

(b) decode $\text{res}(R(2^{n_2+1}))$, which is a corrupted word in the $A_0N$-code, to get $E' = \text{res}(R(2^{n_2+1})) - \text{res}(A_0N)$;

(c) decode $\text{res}(E'/(2^{n_2+1}))$, which is a corrupted word in the $A_0'N$-code, where $A_0' = (2^n - 1)/(2^{n_2} + 1)$, to get $E$.

**Decoding algorithms for Class $C_2$ Codes**

1. If $\ell_1 > \ell_2$,

   (a) $N = 0$ if and only if $W(R) \leq \lfloor (d - 1)/2 \rfloor$, where $d = \min(\ell_2', \ell_1/2)$; otherwise go to (b)

   (b) decode $\text{res}(R(2^{n_2-1}))$, which is a corrupted word in the $A_0N$-code, to get $E' = \text{res}(R(2^{n_2-1})) - \text{res}(A_0N)$;

   (c) decode $\text{res}(E'/(2^{n_2-1}))$, which is a corrupted word in the $A_0'N$-code, where $A_0' = (2^n - 1)/(2^{n_2} - 1)$, to get $E$.

2. If $\ell_2 \geq 2\ell_1 - 1$

   (a) $N = 0$ if and only if $W(R) \leq \lfloor (\ell_1 - 1)/2 \rfloor$; otherwise go to (b)

   (b) decode $\text{res}(R(2^{n_1+1}))$, which is a corrupted word in the $A_0'N$-code where $A_0' = (2^n - 1)/(2^{n_2} - 1)$, to get $E' = \text{res}(R(2^{n_1+1})) - \text{res}(A_0'N)$ [7];
(c) decode \(\text{res} \cdot E'/(2^{n_1+1})\), which is a corrupted word in the \(A_0N\)-code, to get \(E\).

Example 6 illustrates the decoding algorithm for an arithmetic code generated by \(A_0 = (2^n-1)/(2^{n_1} + 1)\), while Example 7 illustrates the decoding algorithm for an arithmetic code in Class \(C_2\).

Example 6: Suppose \(A_0 = (2^{24}-1)/(2^4+1)\), then \(B = 2^4+1\), \(n = 24\) and \(n_1 = 4\). By Theorem 2, \(d = 6\) and this code is capable of correcting any double errors. We have

\[
A_0 = \frac{2^{19} + 2^{18} + 2^{17} + 2^{16} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^3 + 2^2 + 2^1 + 1}{2^{19} + 2^{18} + 2^{17} + 2^{16} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^3 + 2^2 + 2^1 + 1}
= (000011110000111100001111).
\]

If a double error \(E = 2^8 - 2^5\) is added to the codeword \(3A_0\), then the corrupted word is \(R = 3A_0 + E\) with binary representation

\[
R = (001011010010111000001101).
\]

In this case there is a subsequence of the form \(F_1\) in \(R\) with \(n_1+1 = 5\) consecutive 0's. By applying Operation 1, the modified binary representation of \(R\) is \(R' = (0010110101111000101101)\).

We divide \(R'\) into 6 block and complement all the digits in positions \(4i+j\) where \(i = 1, 3, 5\) and \(j = 0, 1, 2, 3\). Then \(R'\) becomes

\[
1101, 1101, 1101, 1110, 1101, 1101.
\]

We check that
Hence
\[ A_0^N = (0010,1101,0010,1101,0010,1101) \]. By subtracting \( A_0^N \) from \( R \), we obtain
\[ E = (0000,0000,0000,0000,0000,0000) = 2^8 - 2^5 \].

Example 7: Suppose \( A = (2^{60} - 1)/(2^6 + 1)(2^{10} - 1) \), then
\[ B = (2^6 + 1)(2^{10} - 1), \ n = 60, \ n_1 = 6 \text{ and } n_2 = 10 \]. By Part (a) of Theorem 4, \( 5 \leq d \leq 6 \) and this code is capable of correcting any double errors. We have
\[ A = \frac{2^{60} - 1}{(2^6 + 1)(2^{10} - 1)} \]
\[ = 2^43 + 2^42 + 2^41 + 2^40 + 2^39 + 2^38 + 2^34 + 2^33 + 2^31 + 2^30 + 2^29 + 2^27 + 2^26 + 2^24 + 2^22 + 2^19 + 2^17 + 2^16 + 2^12 + 2^9 + 2^8 + 2^7 + 2^6 + 1 \]
\[ = (00000000000000001111110001001101100101100010011111000001) \].

It a double error \( E = -2^{53} + 2^{11} \) is added to the codeword 32A, then the corrupted word is \( R = 32A + E \). To decode, we first calculate the residue of \( R(2^{10} - 1) \mod (2^{60} - 1) \) which is
\[ 100000,000000,011111,100000,011111, \]
\[ 100000,100111,011111,111111,011000 \]. (1)

It is found there are one subsequence with more than \( n_1 = 6 \) consecutive 0's and one subsequence with more than \( n_1 \) consecutive 1's. Applying
Operation 1 to (1) yields

\[ 100000,100000,011111,100000,011111,100000,011111,011111,011111,011000 . \]  \hspace{1cm} (2)

We complement all the digits in positions \( 6i+j \) where \( i = 1,3,5,7,9 \) and \( j = 0,1,2,3,4,5 \). Then (2) becomes

\[ 011111,100000,100000,100000,100000,100000,011111,100000,011111,011000 . \]

We check that

<table>
<thead>
<tr>
<th>digits with positions</th>
<th>majority value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2^{6k+5} ), 0 ( \leq ) k ( \leq ) 9</td>
<td>0,1,1,1,1,1,0,0,1,0</td>
</tr>
<tr>
<td>( 2^{6k+4} ), 0 ( \leq ) k ( \leq ) 9</td>
<td>1,0,0,0,0,0,1,1,0,1</td>
</tr>
<tr>
<td>( 2^{6k+3} ), 0 ( \leq ) k ( \leq ) 9</td>
<td>1,0,0,0,0,0,1,1,0,1</td>
</tr>
<tr>
<td>( 2^{6k+2} ), 0 ( \leq ) k ( \leq ) 9</td>
<td>1,0,0,0,0,0,0,1,0,0</td>
</tr>
<tr>
<td>( 2^{6k+1} ), 0 ( \leq ) k ( \leq ) 9</td>
<td>1,0,0,0,0,0,0,1,0,0</td>
</tr>
<tr>
<td>( 2^{6k+0} ), 0 ( \leq ) k ( \leq ) 9</td>
<td>1,0,0,0,0,0,0,1,0,0</td>
</tr>
</tbody>
</table>

Hence the majority decision of \( \text{res}(R(2^{10}-1)) \) yields a codeword

\[ \text{res}(A_0N) = (011111,100000,011111,100000,011111,100000,011111,100000,011111,100000) . \]

The error is now \( E' = \text{res}(R(2^{10}-1)) - \text{res}(A_0N) \), which has the form

\[ 000000,100000,000000,000000,000000,000000,000000,000000,(-1)00000,00(-1)000 . \]

The actual error \( E \) is congruent to \( E'/(2^{10}-1) \mod(2^{60}-1) \)

\[ E(2^{10}-1) \equiv E' = 2^{53} + 2^{43} + 2^{33} + 2^{23} + 2^{13} + 2^{11} + 2^{10} \mod(2^{60}-1) \]

\[ E = 2^{43} + 2^{33} + 2^{23} + 2^{13} + 2^{11} + 2^{10} \mod(2^{60}-1)/(2^{10}-1) \]
which has the binary form
\[0000000000,0000001000,0000001000,0000001000,
0000010001,0000001000 .\] (3)

Again the majority scheme on (3) yields a block 0000001000.
Repeating this block six times, we have
\[0000010000,0000010000,0000010000,0000010000,\]
\[0000010000,0000001000 .\] (4)

The binary integer (4) is a codeword generated by
\(\frac{2^{60}-1}{2^{10}-1}\). Subtracting (4) from (3), we get the actual error E.
\[000000(-1)000,0000000000,0000000000,0000000000,\]
\[0000000100,0000000000 .\]

Hence, the error pattern E is \(-2^{53}+2^{11}\).
V. Discussion

In this paper we have presented a new upper bound on the minimum distance of cyclic arithmetic codes of composite length. This upper bound is quite tight and gives a good estimation of the minimum distance. Two new classes of codes of composite length $n = \ell_1 n_1 = \ell_2 n_2$ have been introduced. The error correction capability of these codes are discussed and in some cases the actual minimum distance is found. Since $n_1$ and $n_2$ need not be relatively prime, some of these new codes have better information rate than the comparable codes found in [7]. Decoding algorithms for these codes have also been provided. They are based on majority-logic decision, and are similar to the decoding algorithm proposed in [7].
Appendix A

In this appendix we will present numerical examples of the application of Theorems 1, 2, 3, and 4. The symbols for the tables are the following:

- \( n \): code length
- \( A \): generator of the code
- \( B \): number of codewords
- \( d \): actual minimum distance
- \( d_{u_1} \): upper bound on \( d \) given by Theorem 1
- \( d_{u_2} \): upper bound on \( d \) given by [11, Theorem 1]
- \( R \): is the code rate \((R = (\log_2 B)/n)\).

Table I gives numerical examples of the application of Theorem 1. The \( d \) of these codes were obtained by a computer search [13].

Table II gives numerical examples of the application of Theorem 2.

Table III and IV give numerical examples of the application of Theorems 3 and 4. In Table III we give upper and lower bounds on \( d \) while in Table IV the actual minimum distance is given.
<table>
<thead>
<tr>
<th>n</th>
<th>A</th>
<th>B</th>
<th>d</th>
<th>d_{u_1}</th>
<th>d_{u_2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>5.7</td>
<td>3.3.13</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>3.257</td>
<td>5.17</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>3.7.19</td>
<td>3.3.73</td>
<td>4</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>5.31</td>
<td>3.5.11.41</td>
<td>3</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>5.31.41</td>
<td>3.5.11</td>
<td>4</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>5.11.31</td>
<td>3.5.41</td>
<td>4</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>3.5.31.41</td>
<td>5.11</td>
<td>6</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>24</td>
<td>3.3.17</td>
<td>5.7.13.241</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>7.17</td>
<td>3.3.5.13.241</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>24</td>
<td>3.3.241</td>
<td>5.7.13.17</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>7.17.241</td>
<td>3.3.5.13</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>24</td>
<td>5.7.241</td>
<td>3.3.13.17</td>
<td>5</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>24</td>
<td>3.3.5.241</td>
<td>7.13.17</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>3.3.13.241</td>
<td>5.7.17</td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>5.7.17.241</td>
<td>3.3.13</td>
<td>6</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>28</td>
<td>5.127</td>
<td>3.29.43.113</td>
<td>3</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>28</td>
<td>29.113.127</td>
<td>3.5.43</td>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>30</td>
<td>7.31.331</td>
<td>9.11.151</td>
<td>6</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>7.31.151.331</td>
<td>9.11</td>
<td>6</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>32</td>
<td>17.65537</td>
<td>3.5.257</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>32</td>
<td>5.65537</td>
<td>3.17.257</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>3.65537</td>
<td>5.17.257</td>
<td>4</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>5.257.65537</td>
<td>3.17</td>
<td>8</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>3.257.65537</td>
<td>5.17</td>
<td>8</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>36</td>
<td>13.73</td>
<td>3.3.3.5.7.19.37.109</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>36</td>
<td>3.19.37.109</td>
<td>3.3.5.7.13.73</td>
<td>3</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>36</td>
<td>5.73</td>
<td>3.3.3.7.13.19.37.109</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>36</td>
<td>3.13.19.73</td>
<td>3.3.5.7.37.109</td>
<td>4</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>n</td>
<td>A</td>
<td>B</td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------------</td>
<td>----</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3·17</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>5·7·13</td>
<td>3·3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>3·17·257</td>
<td>5</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>3·5·257</td>
<td>17</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>3·7·19·73</td>
<td>3·3</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>5·5·31·41</td>
<td>3·11</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>5·7·13·17·241</td>
<td>3·3</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>3·3·5·7·13·241</td>
<td>17</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>3·3·7·17·241</td>
<td>5·13</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>3·29·43·113·127</td>
<td>5</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>5·29·113·127</td>
<td>3·43</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>7·11·31·151·331</td>
<td>3·3</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>3·7·31·151·331</td>
<td>3·11</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>3·5·257·65537</td>
<td>17</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>3·5·17·65537</td>
<td>257</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>3·5·7·13·19·37·73·109</td>
<td>3·3</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>3·3·3·7·19·37·73·109</td>
<td>5·13</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>5·7·13·37·73·109</td>
<td>3·3·3·19</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>B</td>
<td>d&lt;</td>
<td>d&gt;</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---------------------</td>
<td>----</td>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>(2^{15} - 1) (2^{10} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4166</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>(2^{15} + 1) (2^{10} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4167</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>(2^{10} + 1) (2^{6} - 1)</td>
<td>6</td>
<td>5</td>
<td>0.2663</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>(2^{10} - 1) (2^{6} + 1)</td>
<td>6</td>
<td>5</td>
<td>0.2670</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>(2^{18} - 1) (2^{12} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4166</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>(2^{18} + 1) (2^{12} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4166</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>(2^{12} + 1) (2^{9} + 1)</td>
<td>6</td>
<td>4</td>
<td>0.2917</td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>(2^{21} - 1) (2^{14} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4166</td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>(2^{21} + 1) (2^{14} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4166</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{30} - 1) (2^{20} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4166</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{30} + 1) (2^{20} + 1)</td>
<td>4</td>
<td>3</td>
<td>0.4166</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{20} - 1) (2^{12} + 1)</td>
<td>6</td>
<td>5</td>
<td>0.2666</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{20} + 1) (2^{15} + 1)</td>
<td>6</td>
<td>4</td>
<td>0.2916</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{15} - 1) (2^{12} + 1)</td>
<td>8</td>
<td>5</td>
<td>0.2250</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{15} + 1) (2^{12} + 1)</td>
<td>8</td>
<td>5</td>
<td>0.2250</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{12} + 1) (2^{10} - 1)</td>
<td>10</td>
<td>6</td>
<td>0.1833</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>(2^{12} + 1) (2^{10} + 1)</td>
<td>10</td>
<td>6</td>
<td>0.1833</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>B</td>
<td>d=</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------------------------------</td>
<td>----</td>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>((2^{15} - 1)(2^6 + 1))</td>
<td>4</td>
<td>0.3503</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>((2^{15} + 1)(2^6 + 1))</td>
<td>4</td>
<td>0.3503</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>((2^{12} + 1)(2^9 - 1))</td>
<td>6</td>
<td>0.2916</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>((2^{21} - 1)(2^6 + 1))</td>
<td>4</td>
<td>0.3217</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>((2^{21} - 1)(2^6 + 1))</td>
<td>4</td>
<td>0.3217</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>((2^{14} - 1)(2^6 + 1))</td>
<td>6</td>
<td>0.2383</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>((2^{14} + 1)(2^6 - 1))</td>
<td>6</td>
<td>0.2378</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{30} + 1)(2^{12} + 1))</td>
<td>4</td>
<td>0.3500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{30} + 1)(2^{15} - 1))</td>
<td>4</td>
<td>0.3750</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{30} + 1)(2^{15} + 1))</td>
<td>4</td>
<td>0.3750</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{20} + 1)(2^{12} - 1))</td>
<td>6</td>
<td>0.2666</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{20} + 1)(2^{15} - 1))</td>
<td>6</td>
<td>0.2916</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{15} + 1)(2^4 + 1))</td>
<td>8</td>
<td>0.1590</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{15} - 1)(2^6 + 1))</td>
<td>8</td>
<td>0.1751</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{12} + 1)(2^5 - 1))</td>
<td>10</td>
<td>0.1412</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{12} + 1)(2^5 + 1))</td>
<td>10</td>
<td>0.1420</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>((2^{10} + 1)(2^6 - 1))</td>
<td>12</td>
<td>0.1331</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

In this section we will present conditions for the existence of codes in Classes $C_1$ and $C_2$.

Let $k > 1$ be an odd positive integer. $\tilde{e}(k)$ is defined as the least positive integer such that $2^{\tilde{e}(k)} + 1$ is divisible by $k$, if one does exist. $e(k)$ is the exponent of $k$.

At this point we are required to prove the following technical lemmas:

**Lemma B1:** $e(k) = 2\tilde{e}(k)$.

**Proof:** Since $k$ divides $2^{\tilde{e}(k)} + 1$, $k|2^{\tilde{e}(k)} + 1$, then $k|2^{2\tilde{e}(k)} - 1$. Thus $e(k)|2\tilde{e}(k)$. Assume $e(k)$ is odd, then $e(k)|\tilde{e}(k)$. Thus $k|2^{\tilde{e}(k)} - 1$ which is a contradiction since $2^{\tilde{e}(k)} + 1$ and $2^{\tilde{e}(k)} - 1$ are relatively prime. So we can conclude that $e(k)$ is even. Let $e(k) = 2x$, so $x|\tilde{e}(k)$. Assume $\tilde{e}(k) = mx$, $m > 1$. Since $k|(2^x + 1)(2^x - 1)$ and $k|2^x + 1$ there exists $k_1 \neq 1$ such that $k_1|k$ and $k_1|2^x - 1$. Thus, $k_1|2^{\tilde{e}(k)} - 1$ which is a contradiction.

Q.E.D.

**Lemma B2:** $k|2^y + 1$ if and only if $\tilde{e}(k)|y$ and $y/\tilde{e}(k)$ is odd.

**Proof:** Since $(2^y + 1) - (2^{\tilde{e}(k)} + 1) = 2^{\tilde{e}(k)}(2^y - 2^{\tilde{e}(k)} - 1)$, $k|2^y + 1$ if and only if $k|2^y - 2^{\tilde{e}(k)} - 1$, i.e., if and only if $e(k)|y - \tilde{e}(k)$. By Lemma B1, $e(k) = 2\tilde{e}(k)$. So, $k|2^y + 1$ if and only if $\tilde{e}(k)|y$ and $y/\tilde{e}(k)$ is odd.

Q.E.D.
Let \( n = \ell_1 n_1 = \ell_2 n_2, 1 < \ell_1 < n, 1 < \ell_2 < n, m_1 = n_1/g \) and \( m_2 = n_2/g \) where \( g = \text{GCD}(n_1, n_2) \).

We are now in the position to prove the next two theorems which are the main results of this section.

**Theorem B1**: If \( \ell_1, \ell_2 \) and \( m_1 m_2 \) are even integers, then
\[
(2^{\ell_1+1})(2^{\ell_2+1}) \mid 2^n - 1.
\]

**Proof**: By [12, Lemma 6.3], \( 2^{\ell_1+1} \mid 2^n - 1 \) and \( 2^{\ell_2+1} \mid 2^n - 1 \). Now we will show that \( \text{GCD}(2^{\ell_1+1}, 2^{\ell_2+1}) = 1 \). Assume \( a > 1 \) is a common factor of \( 2^{\ell_1+1} \) and \( 2^{\ell_2+1} \). By Lemma B2, \( n_1 = v_1 e(a), n_2 = v_2 e(a) \) with \( v_1 \) and \( v_2 \) odd integers. Thus, \( e(a) \mid g \). So \( m_1 \) divides \( v_1 \) and \( m_2 \) divides \( v_2 \). So, \( m_1 \) and \( m_2 \) are odd which is a contradiction.

**Q.E.D.**

**Theorem B2**: If \( \ell_1 \) is even and \( m_2 \) is odd, then
\[
(2^{\ell_1+1})(2^{n_2-1}) \mid 2^n - 1.
\]

**Proof**: By [12, Lemma 6.3], \( 2^{\ell_1+1} \mid 2^n - 1 \). It is simple to show that \( 2^{n_2-1} \mid 2^n - 1 \). Now we will show that \( \text{GCD}(2^{\ell_1+1}, 2^{n_2-1}) = 1 \). Assume \( a > 1 \) is a common factor of \( 2^{\ell_1+1} \) and \( 2^{n_2-1} \). Then, by Lemma B2, \( n_1 = v_1 e(a) \) with \( v_1 \) odd. By [12, Lemma 6.1], \( n_2 = v_2 e(a) \). By Lemma B1, \( e(a) = 2\tilde{e}(a) \). Since \( m_2 \) is odd, \( g \) must be divisible by \( 2\tilde{e}(a) \). Thus, \( v_1 \) is even which is a contradiction.

**Q.E.D.**
References


