cycamb.tex, 14.2.97

The Generation of Finite Alphabet Codewords With No Mutual Cyclic Shift Equivalence [1]

M.G.Parker, S.J.Shepherd,
Telecommunications Research Group,
Department of Electronic and Electrical Engineering,
University of Bradford,
Bradford, BD7 1DP, UK.
e-mail: `mgparker@bradford.ac.uk`

Abstract

This paper shows how to directly compute a largest set, $\mathbf{C}$*, of length* N *codewords over a given finite alphabet, such that no codeword in* $\mathbf{C}$ *is a cyclic shift of another codeword in* $\mathbf{C}$*. The method uses finite integer rings and can be used to generate irreducible polynomials.*

Definitions: $\lfloor x \rfloor$ is the largest integer value $\leq x$. $\langle a \rangle_m$ means the residue of a, mod m. $\mathrm{ord}_m(a)$ is the order of a, mod m, i.e. $\langle a^i \rangle_m \neq 1$, $0 < i < n$, $\langle a^n \rangle_m = 1$. $\mathrm{Rt}(n, m)$ is an n^{th}root of 1, mod m. Thus $\mathrm{ord}_m(\mathrm{Rt}(n, m)) = n$.

1 Introduction

Consider the finite alphabet, $\mathbf{A}$, and consider the set, $\mathbf{V}$, of all messages, $\mathbf{v}$, of length N over A. Thus,

$$\mathbf{v} \in \mathbf{V}, \qquad \mathbf{v} = (v_0, v_1, \ldots, v_{N-1}), \qquad v_k \in \mathbf{A}, \qquad \forall k$$

Let the f^{th}right cyclic shift operation, $s(\mathbf{v}, f)$, be defined as follows,

$$s(\mathbf{v}, f) = (v_{\langle -f \rangle_N}, v_{\langle 1-f \rangle_N}, \ldots, v_{\langle N-1-f \rangle_N})$$

Consider the codeset $\mathbf{C} \subset \mathbf{V}$, such that,

$$\mathbf{v} \in \mathbf{C} \text{ iff } \mathbf{v}' = s(\mathbf{v}, f) \notin \mathbf{C}, \mathbf{v}' \neq \mathbf{v}, \qquad \forall \mathbf{v} \in \mathbf{V}, 0 < f < N \tag{1}$$

In other words, $\mathbf{C}$ comprises representative members of $\mathbf{V}$, such that $\mathbf{V}$ is generated by the repeated operation of s on $\mathbf{C}$. Moreover, the repeated operation of s on $\mathbf{C}$ never maps a member of $\mathbf{C}$ back into another member of $\mathbf{C}$. The codeset, $\mathbf{C}$, is not uniquely defined by (1) as there are many possible representatives. $\mathbf{C}$ is of interest because there are important functions, H, acting on members of $\mathbf{V}$, which remain invariant under s (shift-invariant), and $\mathbf{C}$ is a smallest subset of $\mathbf{V}$ such that,

$$H(\mathbf{C}) = H(\mathbf{V})$$

where,

$$H(\mathbf{v}) = H(s(\mathbf{v}, f)), \qquad \forall f \tag{2}$$

Moreover,

$$H(\mathbf{c}) \neq H(\mathbf{c}'), \qquad \mathbf{c}, \mathbf{c}' \in \mathbf{C}, \qquad \mathbf{c} \neq \mathbf{c}'$$

[1] *The work described in this paper was supported by EPSRC grant ref: GR/K48914*

For instance, if one enumerates members of $\mathbf{A}$ arbitrarily, and then performs the N-point Discrete Fourier Transform of $\mathbf{v}$, given by,

$$u_n = \sum_{k=0}^{N-1} v_k e^{\frac{2\pi jnk}{N}} = \mathrm{DFT}_n(\mathbf{v})$$

then H could be chosen to satisfy (2) if $H(\mathbf{v}) = \max(|\,\mathrm{DFT}_n(\mathbf{v})|)$. $\mathbf{C}$ would be a useful 'smaller search space' over which to look for 'ideal' channel estimation training sequences [3]. This paper describes a method for computing $\mathbf{C}$ by first mapping $\mathbf{A}$ to the integers, and then using finite integer arithmetic to compute $\mathbf{C}$ [1, 2]. Such a scheme is suitable for software and hardware applications, and also highlights the underlying structure of the message space, $\mathbf{V}$, under cyclic shifts, this being dependent on the factorisation of integers of the form $P^N - 1$.

2 Theory

If $|\mathbf{A}| = P$, $\mathbf{A}$ can be mapped to an integer alphabet, $\mathbf{I_P}$, where $\mathbf{I_P} = \{0, 1, \ldots, P-1\}$. Without loss of generalisation, members $\mathbf{v}$ of $\mathbf{V}$, will forthwith be considered to be messages from the alphabet, $\mathbf{I_P}$. Consider the following bijective mapping,

$$\begin{array}{c} \mathbf{v} \Leftrightarrow w, \quad \forall \mathbf{v} \in \mathbf{V}, \text{ and } w \in \mathbf{Z_M}, \text{ except } \mathbf{v_e} = (P-1, P-1, \ldots, P-1), \\ \text{where } w = \left\langle \sum_{i=0}^{N-1} v_i P^i \right\rangle_M, \quad v_i = \left\langle \lfloor \frac{w}{P^i} \rfloor \right\rangle_P \in \mathbf{I_P} \quad \text{and } M = P^N - 1 \end{array} \tag{3}$$

Both $\mathbf{v} = (0, 0, \ldots, 0)$ and $\mathbf{v_e} = (P-1, P-1, \ldots, P-1)$ map to $w = 0$ under (3) hence the exclusion of $\mathbf{v_e}$. Using the Chinese Remainder Theorem (CRT) [1], each member of $\mathbf{Z_M}$ can be constructed from its R residues over the mutually prime factors of M,

$$\begin{array}{c} \forall w \in \mathbf{Z_M}, \quad w = r_0 \otimes r_1 \otimes \ldots r_{R-2} \otimes r_{R-1}, \quad r_j = \langle w \rangle_{m_j^{t_j}} \\ \Rightarrow w = \left\langle \sum_{j=0}^{R-1} r_j \frac{M}{m_j^{t_j}} \left\langle \left(\frac{M}{m_j^{t_j}} \right)^{-1} \right\rangle_{m_j^{t_j}} \right\rangle_M \end{array} \tag{4}$$

where $M = \prod_{j=0}^{R-1} m_j^{t_j}$, and "$\otimes$" means the direct product. Let $n_{e_j,0,j} = \mathrm{ord}_{m_j^{e_j}}(P)$, and $n_{e_j,1,j} = \frac{\phi'(m_j^{e_j})}{n_{e_j,0,j}}$, where,

$$\begin{array}{ll} \phi'(m_j^{e_j}) = \phi(m_j^{e_j}) & m_j \neq 2 \text{ and/or } e_j \leq 2 \\ \phi'(m_j^{e_j}) = \frac{\phi(m_j^{e_j})}{2} & m_j = 2 \text{ and } e_j > 2 \end{array}$$

and ϕ is Euler's Totient Function [1]. If $\gcd(n_{e_j,0,j}, n_{e_j,1,j}) = 1$, let $\beta_{e_j,j} = \mathrm{Rt}(n_{e_j,1,j}, m_j^{e_j})$, (a prime factor combination of the exponents, n). Alternatively, or if $\gcd(n_{e_j,0,j}, n_{e_j,1,j}) > 1$, let $\beta_{e_j,j} = \mathrm{Rt}(\phi'(m_j^{e_j}), m_j^{e_j})$, (a mixed-radix combination of the exponents, n). For all cases except $m_j = 2$, $e_j > 2$, each residue, r_j, can be generated using the following construction,

$$r_j \in \left\{ \left\langle m_j^{t_j - e_j} P^{s_{e_j,0,j}} \beta_{e_j,j}^{s_{e_j,1,j}} \right\rangle_{m_j^{t_j}} \quad , \quad 0 \quad : \quad 1 \leq e_j \leq t_j, 0 \leq s_{e_j,i,j} < n_{e_j,i,j}, i \in \{0, 1\} \right\} \tag{5}$$

When $m_j = 2$ and $e_j > 2$, r_j is generated by,

$$\begin{array}{c} r_j \in \left\{ \left\langle 2^{t_j - e_j} P^{s_{e_j,0,j}} \beta_{e_j,j}^{s_{e_j,1,j}} \right\rangle_{2^{t_j}} \quad , \quad \left\langle \mu 2^{t_j - e_j} P^{s_{e_j,0,j}} \beta_{e_j,j}^{s_{e_j,1,j}} \right\rangle_{2^{t_j}} \quad , \right. \\ \left. 0 \quad : \quad 2 \leq e_j \leq t_j, 0 \leq s_{e_j,i,j} < n_{e_j,i,j}, i \in \{0, 1, \} \right\} \end{array} \tag{6}$$

where μ is given by,

$$\mu = -1 \qquad \text{if } \langle P+1\rangle_{2^{e_j}} \neq 0$$
$$\mu = 2^{e_j-1}+1 \qquad \text{if } \langle P+1\rangle_{2^{e_j}} = 0$$

By ranging through all possible values of e_j and $s_{e_j,i,j}$ for a given j, (5) and (6) generate the $m_j^{t_j}$ integers, $\{1,\ldots,m_j^{t_j}-1\}+\{0\}$. The generation of all $w \in \mathbf{Z_M}$, (and therefore all $\mathbf{v} \in \mathbf{V}, \mathbf{v} \neq \mathbf{v_e}$), is achieved by constructing the r_j with (5) and/or (6), and then using (4) to form each w. To generate only codewords, $\mathbf{c} \in \mathbf{C}$, the criteria of (5), (6), are modified. Consider the operation $s(\mathbf{v},1)$. This is equivalent to the operation $\langle wP\rangle_M$ which, in turn, is equivalent to, $\left(\langle r_0P\rangle_{m_0^{t_0}} \otimes \langle r_1P\rangle_{m_1^{t_1}} \otimes \ldots \otimes \langle r_{R-1}P\rangle_{m_{R-1}^{t_{R-1}}}\right)$. From (5) and (6), the operation $\langle r_jP\rangle_{m_j^{t_j}}$ is achieved by replacing $s_{e_j,0,j}$ with $\left\langle s_{e_j,0,j}+1\right\rangle_{n_{e_j,0,j}}$, $\forall e_j, j$, with $s_{e_j,1,j}$ unchanged. Thus, to generate w_c corresponding to all codewords, $\mathbf{c} \in \mathbf{C}$, the $n_{e_j,1,j}$ in (5) and (6) are left unchanged, whereas the $n_{e_j,0,j}$ are replaced by $n'_{e_j,0,j,q}$, $\forall q$, $0 \leq q < Q$, where q is constructed using the following mixed-radix formulation,

$$q = \sum_{j=0}^{R-1} e_j \prod_{i=0}^{j-1}(t_i+1) \qquad 0 \leq e_j \leq t_j, \qquad\qquad Q = \prod_{j=0}^{R-1}(t_j+1)$$

The $n'_{e_j,0,j,q}$ are evaluated as follows,

$$n'_{e_j,0,j,q} = \gcd(\mathrm{lcm}(\gamma(n_{e_{j+1},0,j+1}), \gamma(n_{e_{j+2},0,j+2}), \ldots \gamma(n_{e_{R-1},0,R-1}), 1), n_{e_j,0,j}) \tag{7}$$

where $n_{0,0,j} = 0$, $\forall j$, and $\gamma(n) = n$, $n > 0$, $\gamma(0) = 1$. The $n_{0,0,j}$ are not required in (5) or (6) (as e_j is never zero), so their replacements, $n'_{0,0,j,q}$, need not be computed in (7). The values, $n_{0,0,j} = 0$ are only included in (7) for $r_j = 0$. For each q, the $n'_{e_j,0,j,q}$ can be computed using (7), to replace the respective $n_{e_j,0,j}$ in (5) or (6), and a subset of $\mathbf{C}$, $\mathbf{C_q}$, can be generated, using (5) and/or (6), (4), and (3). Thus,

$$\mathbf{C} = \left(\cup_{q=0}^{Q-1}\mathbf{C_q}\right) \cup \mathbf{v_e}$$

3 Examples

Example 1: Let $P = 2$, $N = 5$. Then $M = 2^5 - 1 = 31$, and 31 is prime. Thus $R = 1$, $m_0 = 31$, $t_0 = 1$, and $r_0 = \{\langle 2^{s_{e_0,0,0}} 6^{s_{e_0,1,0}}\rangle_{31} \quad , \quad 0 : \quad 1 \leq e_0 \leq 1, 0 \leq s_{e_0,i,0} < n_{e_0,i,0}\}$, where $n_{e_0,0,0} = 5$, $n_{e_0,1,0} = 6$. To generate w_c corresponding to all codewords, $\mathbf{c} \in \mathbf{C}$, $n_{e_0,0,0}$ is limited to $n'_{e_0,0,0,q}$, $\forall q$, $0 \leq q < 2$, as follows,

$$n'_{0,0,0,0} \qquad \text{is not required.}$$
$$n'_{1,0,0,1} = \gcd(\mathrm{lcm}(1), n_{1,0,0}) = 1$$

With $n'_{1,0,0,1} = 1$, $r_0 \in \{2^06^0, 2^06^1, 2^06^2, 2^06^3, 2^06^4, 2^06^5, 0\} = \{1, 6, 5, 30, 25, 26, 0\}$. The CRT construction is trivial, i.e. $w_c = r_0$, $\forall r_0$. The 8 codewords, $\mathbf{c} \in \mathbf{C}$, are,

$0,0,0,0,1 \quad (w_c = 1)$, $\quad 0,0,1,1,0 \quad (w_c = 6)$, $\quad 0,0,1,0,1 \quad (w_c = 5)$,
$1,1,1,1,0 \quad (w_c = 30)$, $\quad 1,1,0,0,1 \quad (w_c = 25)$, $\quad 1,1,0,1,0 \quad (w_c = 26)$,
$0,0,0,0,0 \quad (w_c = 0)$, $\quad 1,1,1,1,1 \quad$ (Exception $= 31$)

Example 2: Let $P = 15$, $N = 2$. Then $M = 15^2 - 1 = 224 = 2^5 7$. Thus $R = 2$, $m_0 = 2$, $m_1 = 7$, $t_0 = 5$, and $t_1 = 1$.

$$
\begin{array}{rcl}
r_0 = \{\langle 16.15^{s_{1,0,0}} 1^{s_{1,1,0}} \rangle_{32} & , & \\
\langle 8.15^{s_{2,0,0}} 1^{s_{2,1,0}} \rangle_{32} & , & \\
\langle 4.15^{s_{3,0,0}} 7^{s_{3,1,0}} \rangle_{32} \quad , & \langle 5.4.15^{s_{3,0,0}} 7^{s_{3,1,0}} \rangle_{32} & , \\
\langle 2.15^{s_{4,0,0}} 3^{s_{4,1,0}} \rangle_{32} \quad , & \langle 9.2.15^{s_{4,0,0}} 3^{s_{4,1,0}} \rangle_{32} & , \\
\langle 15^{s_{5,0,0}} 3^{s_{5,1,0}} \rangle_{32} \quad , & \langle -15^{s_{5,0,0}} 3^{s_{5,1,0}} \rangle_{32} & , \\
0 \quad : & 0 \le s_{e_0,i,0} < n_{e_0,i,0} \} &
\end{array}
$$

where $n_{1,0,0} = 1, n_{2,0,0} = 2, n_{3,0,0} = 2, n_{4,0,0} = 2, n_{5,0,0} = 2$, and $n_{1,1,0} = 1, n_{2,1,0} = 1, n_{3,1,0} = 1, n_{4,1,0} = 2, n_{5,1,0} = 4$. For $e_j > 2$, (6) is used instead of (5). For $e_j = 3$ and 4, $\mu = 5$ and 9, respectively. For $e_j = 5$, $\mu = -1$.

$$
r_1 = \{\langle 15^{s_{1,0,1}} 3^{s_{1,1,1}} \rangle_7 \quad , \quad 0 \quad : \quad 0 \le s_{1,i,1} < n_{1,i,1}\}
$$

where $n_{1,0,1} = 1$ and $n_{1,1,1} = 6$. To generate w_c, corresponding to all codewords, $\mathbf{c} \in \mathbf{C}$, the $n_{e_j,0,j}$ are limited to $n'_{e_j,0,j,q}$, $\forall q$, $0 \le q < Q$, where $Q = 6.2 = 12$, as follows,

$$
\begin{array}{rr}
n'_{0,0,0,0} \text{ not required }, & n'_{0,0,1,0} \text{ not required} \\
n'_{1,0,0,1} = 1, & n'_{0,0,0,1} \text{ not required} \\
n'_{2,0,0,2} = 1, & n'_{0,0,0,2} \text{ not required} \\
n'_{3,0,0,3} = 1, & n'_{0,0,0,3} \text{ not required} \\
n'_{4,0,0,4} = 1, & n'_{0,0,0,4} \text{ not required} \\
n'_{5,0,0,5} = 1, & n'_{0,0,0,5} \text{ not required} \\
n'_{0,0,0,6} \text{ not required}, & n'_{1,0,0,6} = 1 \\
n'_{1,0,0,7} = 1, & n'_{1,0,0,7} = 1 \\
n'_{2,0,0,8} = 1, & n'_{1,0,0,8} = 1 \\
n'_{3,0,0,9} = 1, & n'_{1,0,0,9} = 1 \\
n'_{4,0,0,10} = 1, & n'_{1,0,0,10} = 1 \\
n'_{5,0,0,11} = 1, & n'_{1,0,0,11} = 1
\end{array}
$$

For each q, $0 \le q < 12$, 119 different (r_0, r_1) residue pairs are generated:

$q = 0 : (0, 0);$
$q = 1 : (16, 0);$
$q = 2 : (8, 0);$
$q = 3 : (4, 0), (20, 0);$
$q = 4 : (2, 0), (2.3, 0), (9.2, 0), (9.2.3, 0);$
$q = 5 : (1, 0), (1.3, 0), (1.3^2, 0), (1.3^3, 0), (-1, 0), (-1.3, 0), (-1.3^2, 0), (-1.3^3, 0);$
$q = 6 : (0, 1), (0, 1.3), (0, 1.3^2), (0, 1.3^3), (0, 1.3^4), (0, 1.3^5);$
$q = 7 : (16, 1), (16, 1.3), (16, 1.3^2), (16, 1.3^3), (16, 1.3^4), (16, 1.3^5);$
$q = 8 : (8, 1), (8, 1.3), (8, 1.3^2), (8, 1.3^3), (8, 1.3^4), (8, 1.3^5);$
$q = 9 : (4, 1), (4, 1.3), (4, 1.3^2), (4, 1.3^3), (4, 1.3^4), (4, 1.3^5), (20, 1), (20, 1.3), (20, 1.3^2), (20, 1.3^3), (20, 1.3^4), (20, 1.3^5);$
$q = 10 : (2, 1), (2, 1.3), (2, 1.3^2), (2, 1.3^3), (2, 1.3^4), (2, 1.3^5), (2.3, 1), (2.3, 1.3), (2.3, 1.3^2), (2.3, 1.3^3), (2.3, 1.3^4), (2.3, 1.3^5),$
$(9.2, 1), (9.2, 1.3), (9.2, 1.3^2), (9.2, 1.3^3), (9.2, 1.3^4), (9.2, 1.3^5), (9.2.3, 1), (9.2.3, 1.3), (9.2.3, 1.3^2), (9.2.3, 1.3^3),$
$(9.2.3, 1.3^4), (9.2.3, 1.3^5);$
$q = 11 : (1, 1), (1, 1.3), (1, 1.3^2), (1, 1.3^3), (1, 1.3^4), (1, 1.3^5), (1.3, 1), (1.3, 1.3), (1.3, 1.3^2), (1.3, 1.3^3), (1.3, 1.3^4), (1.3, 1.3^5),$
$(1.3^2, 1), (1.3^2, 1.3), (1.3^2, 1.3^2), (1.3^2, 1.3^3), (1.3^2, 1.3^4), (1.3^2, 1.3^5), (1.3^3, 1), (1.3^3, 1.3), (1.3^3, 1.3^2), (1.3^3, 1.3^3),$
$(1.3^3, 1.3^4), (1.3^3, 1.3^5), (-1, 1), (-1, 1.3), (-1, 1.3^2), (-1, 1.3^3), (-1, 1.3^4), (-1, 1.3^5), (-1.3, 1), (-1.3, 1.3),$
$(-1.3, 1.3^2), (-1.3, 1.3^3), (-1.3, 1.3^4), (-1.3, 1.3^5), (-1.3^2, 1), (-1.3^2, 1.3), (-1.3^2, 1.3^2), (-1.3^2, 1.3^3), (-1.3^2, 1.3^4),$
$(-1.3^2, 1.3^5), (-1.3^3, 1), (-1.3^3, 1.3), (-1.3^3, 1.3^2), (-1.3^3, 1.3^3), (-1.3^3, 1.3^4), (-1.3^3, 1.3^5);$

Using the CRT, $w_c = \langle 161 r_0 + 64 r_1 \rangle_{224}$. The 119 integers, w_c, each corresponding to a codeword, $c \in \mathbf{C}$, can be computed from the above.

4 The Generation of Irreducible Polynomials

This section describes how to use the above method to generate irreducible polynomials. If $\beta \in \text{GF}(p^N)$, $\beta \notin \text{GF}(p^n)$, $n|N$, $n \ne N$, then $\beta^{p^0}, \beta^{p^1}, \ldots, \beta^{p^{N-1}}$ is a normal basis for $\text{GF}(p^N)$. Moreover, the N conjugates of β are roots of a degree N irreducible polynomial, $I(x)$, over $\text{GF}(p)$,

$$
I(x) = \prod_{i=0}^{N-1} (x - \beta^{p^i})
$$

Let us represent β^k using the normal basis in β. Then successive cyclic shifts, s, of the basis coefficients generate all conjugates of β^k. If $s^n(\beta^k) = \beta^k$, $s^{n'}(\beta^k) \neq \beta^k$, $0 < n' < n$, then β^k has n conjugates, where $n|N$, and the n conjugates of β are roots of a degree n irreducible polynomial, $I_k(x)$, over $\mathrm{GF}(p)$,

$$I_k(x) = \prod_{i=0}^{n-1}(x - \beta^{kp^i})$$

Therefore each irreducible polynomial, $I_k(x)$, of degree n, $n|N$, can be uniquely associated with one of its roots, β^{kp^i}, and the generation of a maximum subset, $\mathbf{C}$, of length N words over an integer alphabet, $\mathbf{I_p}$, such that no word is a cyclic shift of another, is equivalent to the generation of a representative root of all possible $I_k(x)$ of degree n, $n|N$. $|\mathbf{C}|$ is equal to the number of irreducible polynomials over $\mathrm{GF}(p)$ of degree n, $n|N$. This paper has shown how to directly compute $\mathbf{C}$ and this section has shown that, when $P = p$ is a power of a prime, each member of $\mathbf{C}$ is a root of a different irreducible polynomial, $I_k(x)$, when interpreted over a normal basis. Thus the method of this paper can be used to generate all possible irreducible polynomials over a given field. It is straightforward to extend the technique to all integer values of P by use of Residue Number Systems [2]. From [1] it is known that the number of irreducible polynomials over $\mathrm{GF}(p)$ of degree n, $n|N$, satisfies,

$$\sum_{n|N} \frac{1}{n} \sum_{d|n} \mu(d) p^{\frac{n}{d}} \tag{8}$$

where μ is the Mobius Function. This can be used to verify the method of this paper for the cases where $P = p$ is a power of a prime. For instance, when $p = 2$, $N = 5$, then (8) gives 8 irreducible polynomials, and this is verified by Example 1.

5 Conclusion

This paper has described a method for the generation of a largest subset of length N messages over a finite alphabet so that no two messages in the subset are equivalent under cyclic shift. The method requires finite integer arithmetic. A similar method would use finite polynomial arithmetic to achieve the same ends. It has also been shown how the method can be used to generate all irreducible polynomials over a given field.

References

[1] R.Lidl,H.Niederreiter, **Introduction to Finite Fields and their Applications,** Cambridge Univ Press, '86

[2] M.G.Parker, "VLSI Algorithms and Architectures for the Implementation of Number-Theoretic Transforms, Residue and Polynomial Residue Number Systems," *PhD thesis, School of Eng, University of Huddersfield, March '95*

[3] C.Tellambura,M.G.Parker,Y.J.Guo,S.J.Shepherd,S.K.Barton, "Optimal Sequences for Channel Estimation Using Discrete Fourier Transform Techniques," *Accepted for Publication in IEEE Trans on Communications, '96*