

Wyner-Ziv bound for the rate distortion function  $R(D)$  was applied to this class of source and was shown as follows:

$$R(D) \geq R_{LB}(D) = R_1(D) + \frac{1}{4\pi} \int_{-\pi}^{\pi} \log S_n(\omega) d\omega,$$

where  $R_1(D)$  is the rate distortion function for the corresponding memoryless source and  $S_n(\omega)$  is the power spectral density of the underlying Gaussian source. The second term in the equation reflects the degree of memory of the source measured by the degree of spectral flatness. It is zero iff the source is memoryless, i.e., the spectrum becomes completely flat. The utility of this lower bound is that it permits us to bound the rate distortion function by two terms and each of which is relatively easy to calculate. The first term, which depends only on the memoryless nonlinearity, can be easily computed using the Blahut algorithm [8] which applies to any first-order distribution. This result gives  $R_1(D)$  which is also the upper bound on  $R(D)$ . The second term reflects the memory properties of the underlying Gaussian process and is independent of the nonlinear transformation. This term is only involved in a simple integral and is also easy to compute. When this term is small compared to  $R_1(D)$ , which is the case for small  $D$ , the bound is tight. This has practical applications since it is possible to ascertain the extent to which data compression efficiency is influenced by the degree of source correlation. Several examples and figures were illustrated to verify the tightness and usefulness of the lower bound.

#### ACKNOWLEDGMENT

Authors would like to thank Dr. A. Wyner from AT&T Bell Laboratories for his comments on the correspondence. Constructive comments and suggestions from the editor and the associated reviewers have also significantly improved the presentation of this material.

#### REFERENCES

- [1] C. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat. Conv. Rec.*, pp. 142-163, 1959.
- [2] T. Berger, *Rate Distortion Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1971.
- [3] J. Binia, M. Zakai, and J. Ziv, "On the  $\epsilon$ -entropy and rate distortion function of certain non-Gaussian processes," *IEEE Trans. Inform. Theory*, vol. IT-20, no. 4, July 1974.
- [4] A. Wyner and J. Ziv, "Bounds on the rate-distortion function for stationary sources with memory," *IEEE Trans. Inform. Theory*, vol. IT-17, no. 9, pp. 508-513, Sept. 1971.
- [5] H. Derin, "The use of Gibbs distributions in image processing," in *Communications and Networks: A Survey of Recent Advances*, I. Block and V. Poor, Eds. New York: Springer, 1985.
- [6] B. Liu and D. Munson, "Generation of a random sequence having a jointly specified marginal distribution and autocovariance," *IEEE Trans. Acoustics Speech Signal Processing*, vol. 30, no. 6, pp. 973-983, Dec. 1982.
- [7] Y.-Q. Zhang, M. Loew, and R. Pickholtz, "Probabilistic image models and their information-theoretic properties," in *IEEE Trans. Signal Processing*, vol. 41, no. 1, pp. 508-512, Jan. 1993.
- [8] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 7, pp. 460-473, July 1972.
- [9] A. Viterbi and J. Omura, *Principles of Digital Communication and Coding*. New York: McGraw-Hill, 1979, ch. 7.
- [10] R. Rao and W. Pearlman, "On entropy of pyramid structures," *IEEE Trans. Inform. Theory*, vol. 37, no. 2, pp. 407-413, Mar. 1991.

## Weight Enumerators of More Irreducible Cyclic Binary Codes

Robert L. Ward

**Abstract**—The weight enumerators of all nondegenerate irreducible cyclic binary  $(n, k)$ -codes have been computed for which  $k > 27$  and

$N = (2^k - 1)/n < 500$ . The methods used are those developed by McEliece and further expanded by Ward and Segal, using algebraic number theory and Stickelberger's theorem.

**Index Terms**—Cyclic codes, error-correcting codes, algebraic number theory, weight enumerator, Stickelberger's theorem.

#### I. INTRODUCTION

The weight enumerator of an irreducible cyclic binary code can be determined by a method described in [1]. Another method, essentially exhaustion, was described in [2]. This correspondence is an application of the former method to determine the weight enumerators of codes not previously determined.

We will use the notation of [1]. We are given a prime  $p$  and a natural number  $N$  not a multiple of  $p$ , and let  $k$  be the order of  $p$  modulo  $N$ , denoted  $k = \text{ord}_N p$ . Set  $q = p^k$  and  $n = (q - 1)/N$ . Then there is an irreducible polynomial of degree  $k$  over  $\text{GF}(p)$ , which has  $N$  cycles of length  $n$  (and the zero cycle of length 1). These cycles are equivalence classes of  $n$ -tuples under the equivalence relation generated by the cyclic shift operator. These  $n$ -tuples form an  $(n, k)$  irreducible cyclic linear error-correcting code. We wish to find the weight enumerator of the code, or the weights of the cycles.

Let  $\zeta = e^{2\pi i/p}$ , and let  $\psi$  be a primitive root in  $\text{GF}(q)$ . Then set

$$\eta_i = \sum_{j=1}^{n-1} \zeta^{T(\psi^{i+jN})},$$

where  $T$  is the trace map from  $\text{GF}(q)$  to  $\text{GF}(p)$ ,

$$T(\xi) = \sum_{r=0}^{k-1} \xi^{p^r}.$$

Notice that  $i \equiv i' \pmod{N}$  implies that  $\eta_i = \eta_{i'}$ . The weight enumerator of the code is completely determined by the generating function

$$H(x) = \sum_{i=0}^{N-1} \eta_i x^i.$$

Furthermore, the weight of any codeword in class  $i$  is given by  $W_i = (n - \eta_i)(p - 1)/p$ , and the weight enumerator polynomial will have the form

$$w(z) = 1 + n \sum_{i=0}^{N-1} z^{W_i}.$$

Manuscript received July 30, 1992. This work was presented in part at the Annual Winter Meetings of the American Mathematical Society, Baltimore, MD, January 1992.

The author is with the National Security Agency. His home address is 12236 Shadetree Lane, Laurel, MD 20708-2832.

IEEE Log Number 9211274.

## II. A PARTICULAR EXAMPLE

We choose the values  $p = 2$  and  $N = 215$ . Then  $k = 28$ ,  $q = 2^{28} = 268435456$ , and  $n = 1248537$ . Let  $\beta = e^{2\pi i/215}$ . Its minimum polynomial is  $\Phi_{215}(x)$ , which has degree  $\phi(215) = 168$ . The fixed field  $\Omega$  of the automorphism  $\lambda_2$ , defined by  $\lambda_2(\beta) = \beta^2$  and extending linearly, is of index  $k = 28$  in  $\mathbb{Q}(\beta)$ , and so has degree  $K = 168/28 = 6$ .

The next step is to explicitly construct the Galois group, which we know is abelian of order 6, and so is cyclic. The automorphisms of  $\mathbb{Q}(\beta)$  are of the form  $\lambda_a$ , defined by  $\lambda_a(\beta) = \beta^a$  and extending linearly, where  $(a, 215) = 1$ . Since  $\lambda_a[\lambda_b(\beta)] = \lambda_{ab}(\beta)$ , the Galois group is isomorphic to the multiplicative group of units modulo 215 modulo its cyclic subgroup generated by 2. The cosets have least representatives 1, 3, 7, 13, 21, and 39, and  $\sigma = \lambda_{13}$  is a generator of the Galois group.

We next construct the lattice of fields contained in  $\Omega$  and containing  $\mathbb{Q}$ . Since the Galois group has only one subgroup of order 2 and one of order 3, there are unique subfields of  $\Omega$  of degrees 3 and 2, respectively. Since the Jacobi symbols  $(\frac{2}{5}) = -1$ ,  $(\frac{2}{43}) = -1$ , and  $(\frac{2}{215}) = 1$ , the unique quadratic subfield must be  $\mathbb{Q}(\sqrt{-215}) = \mathbb{Q}(\omega)$ , where  $\omega = (1 + \sqrt{-215})/2$ . We introduce  $\omega$  satisfying  $\omega^2 - \omega + 54 = 0$  here because  $\{1, \omega\}$  forms an integral basis of the ring of integers  $O_{\mathbb{Q}(\sqrt{-215})}$ . An element in  $\Omega$  is  $\gamma = \text{tr}_{\mathbb{Q}(\beta)/\Omega}(\beta)$ . This satisfies the minimal polynomial

$$\gamma^6 - \gamma^5 + 47\gamma^4 + 109\gamma^3 + 712\gamma^2 + 652\gamma + 1936 = 0.$$

The element  $\alpha = \gamma \cdot \lambda_{39}(\gamma)$  is fixed by the involution  $\lambda_{39}$ , which is complex conjugation on  $\Omega$ , so lies in the cubic subfield. It satisfies the minimal polynomial  $\alpha^3 - 61\alpha^2 + 710\alpha - 1936 = 0$ . An integral basis for  $O_{\mathbb{Q}(\alpha)}$  is given by  $\{1, \alpha, (\alpha^2 + 109\alpha + 30)/226\}$ . We wish to find an integral basis of  $O_\Omega$ . We quickly find that

$$\{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5\} \\ = \left\{ 1, \gamma, \frac{\gamma^2 + \gamma}{2}, \frac{\gamma^3 + \gamma}{2}, \frac{\gamma^4 + 10\gamma^3 + 3\gamma^2 + 18\gamma + 16}{24}, \right. \\ \left. \frac{\gamma^5 + 144\gamma^4 + 1127\gamma^3 + 372\gamma^2 + 1588\gamma + 4400}{6336} \right\}$$

is such a basis.

Next we consider the action of the automorphisms on the integral basis:

$$\begin{aligned} \sigma(\theta_0) &= \theta_0 \\ \sigma(\theta_1) &= -19\theta_0 + 9\theta_1 + 4\theta_2 + 8\theta_3 - 38\theta_4 + 69\theta_5 \\ \sigma(\theta_2) &= 4\theta_0 - 3\theta_1 + 5\theta_2 - \theta_3 - \theta_4 + 4\theta_5 \\ \sigma(\theta_3) &= 317\theta_0 - 118\theta_1 - 31\theta_2 - 121\theta_3 + 544\theta_4 - 974\theta_5 \\ \sigma(\theta_4) &= 305\theta_0 - 119\theta_1 - 54\theta_2 - 127\theta_3 + 593\theta_4 - 1071\theta_5 \\ \sigma(\theta_5) &= 134\theta_0 - 53\theta_1 - 27\theta_2 - 57\theta_3 + 269\theta_4 - 487\theta_5. \end{aligned}$$

We also consider the multiplication in the ring  $O_\Omega$  in terms of the

integral basis elements:

$$\begin{aligned} \theta_0\theta_0 &= \theta_0 \\ \theta_0\theta_1 &= \theta_1 \\ \theta_0\theta_2 &= \theta_2 \\ \theta_0\theta_3 &= \theta_3 \\ \theta_0\theta_4 &= \theta_4 \\ \theta_0\theta_5 &= \theta_5 \\ \theta_1\theta_1 &= -\theta_1 + 2\theta_2 \\ \theta_1\theta_2 &= \theta_3 \\ \theta_1\theta_3 &= -8\theta_0 - 3\theta_1 - 2\theta_2 - 10\theta_3 + 12\theta_4 \\ \theta_1\theta_4 &= -94\theta_0 + 24\theta_1 + 4\theta_2 + 18\theta_3 - 134\theta_4 + 264\theta_5 \\ \theta_1\theta_5 &= -51\theta_0 + 14\theta_1 + 2\theta_2 + 11\theta_3 - 75\theta_4 + 145\theta_5 \\ \theta_2\theta_2 &= -4\theta_0 - 2\theta_1 - \theta_2 - 4\theta_3 + 6\theta_4 \\ \theta_2\theta_3 &= -528\theta_0 + 156\theta_1 + 29\theta_2 + 152\theta_3 - 858\theta_4 + 1584\theta_5 \\ \theta_2\theta_4 &= -553\theta_0 + 164\theta_1 + 6\theta_2 + 167\theta_3 - 881\theta_4 + 1584\theta_5 \\ \theta_2\theta_5 &= -242\theta_0 + 72\theta_1 + 74\theta_2 - 384\theta_3 + 685\theta_4 \\ \theta_3\theta_3 &= -828\theta_0 + 256\theta_1 - 258\theta_2 + 327\theta_3 - 1134\theta_4 + 1584\theta_5 \\ \theta_3\theta_4 &= 1153\theta_0 - 332\theta_1 - 358\theta_2 - 247\theta_3 + 2005\theta_4 - 4224\theta_5 \\ \theta_3\theta_5 &= 760\theta_0 - 220\theta_1 - 168\theta_2 - 180\theta_3 + 1278\theta_4 - 2591\theta_5 \\ \theta_4\theta_4 &= 3256\theta_0 - 946\theta_1 - 376\theta_2 - 862\theta_3 + 5255\theta_4 - 10120\theta_5 \\ \theta_4\theta_5 &= 1701\theta_0 - 494\theta_1 - 166\theta_2 - 457\theta_3 + 2725\theta_4 - 5200\theta_5 \\ \theta_5\theta_5 &= 868\theta_0 - 252\theta_1 - 72\theta_2 - 236\theta_3 + 1382\theta_4 - 2617\theta_5. \end{aligned}$$

Let  $M$  be the 6-by-6-by-6 three-dimensional tensor whose planes are the following six matrices:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & 0 & -8 & -94 & -51 \\ 1 & -1 & -1 & -3 & 24 & 14 \\ 0 & 2 & 1 & -2 & 4 & 2 \\ 0 & 0 & 1 & -10 & 18 & 11 \\ 0 & 0 & 0 & 12 & -134 & -75 \\ 0 & 0 & 0 & 0 & 264 & 145 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 0 & -4 & -528 & -553 & -242 \\ 0 & -1 & -2 & 156 & 164 & 72 \\ 1 & 1 & -1 & 29 & 6 & 0 \\ 0 & 1 & -4 & 152 & 167 & 74 \\ 0 & 0 & 6 & -858 & -881 & -384 \\ 0 & 0 & 0 & 1584 & 1584 & 685 \end{bmatrix},$$

$$\begin{bmatrix} 0 & -8 & -528 & -828 & 1153 & 760 \\ 0 & -3 & 156 & 256 & -332 & -220 \\ 0 & -2 & 29 & -258 & -358 & -168 \\ 1 & -10 & 152 & 327 & -247 & -180 \\ 0 & 12 & -858 & -1134 & 2005 & 1278 \\ 0 & 0 & 1584 & 1584 & -4224 & -2591 \end{bmatrix},$$

$$\begin{bmatrix} 0 & -94 & -553 & 1153 & 3256 & 1701 \\ 0 & 24 & 164 & -332 & -946 & -494 \\ 0 & 4 & 6 & -358 & -376 & -166 \\ 0 & 18 & 167 & -247 & -862 & -457 \\ 1 & -134 & -881 & 2005 & 5255 & 2725 \\ 0 & 264 & 1584 & -4224 & -10120 & -5200 \end{bmatrix}.$$

$$\begin{bmatrix} 0 & -51 & -242 & 760 & 1701 & 868 \\ 0 & 14 & 72 & -220 & -494 & -252 \\ 0 & 2 & 0 & -168 & -166 & -72 \\ 0 & 11 & 74 & -180 & -457 & -236 \\ 0 & -75 & -384 & 1278 & 2725 & 1382 \\ 1 & 145 & 685 & -2591 & -5200 & -2617 \end{bmatrix}.$$

Then multiplication of two integral elements expressed as linear combinations of the integral basis can be accomplished by multiplying their coefficient vectors  $x$  and  $y$  times  $M$ , one on either side. The resulting vector  $xMy^T$  is the vector of coefficients of the product with respect to the integral basis.

The norm of an element whose vector of coefficients (with respect to the integral basis) is  $x$ , is easily defined in terms of  $M$  as

$$\text{Norm}(x) = \det(xM) = \det(Mx^T).$$

We search for elements with small coefficients such that the norm of these elements is divisible by only small primes, say up to 3. We find the following vectors of coefficients:

$(-1, 1, 0, 0, 0, 0),$	$\text{Norm} = 2^7 \cdot 3^3$
$(2, 1, 0, 0, 0, 0),$	$\text{Norm} = 2^7 \cdot 3^3$
$(-1, 0, 1, 0, 0, 0),$	$\text{Norm} = 2^8 \cdot 3^6$
$(2, 0, 1, 0, 0, 0),$	$\text{Norm} = 2^5 \cdot 3^3$
$(3, 0, 1, 0, 0, 0),$	$\text{Norm} = 2^{12}$
$(1, -3, 2, 0, 0, 0),$	$\text{Norm} = 2^{14} \cdot 3^6$
$(-2, -2, -1, 1, 0, 0),$	$\text{Norm} = 2^{15} \cdot 3^9$
$(1, -2, 0, 1, 0, 0),$	$\text{Norm} = 2^{15} \cdot 3^9$
$(-2, 1, 0, 1, 0, 0),$	$\text{Norm} = 2^{12} \cdot 3^6$
$(-3, 2, 0, 1, 0, 0),$	$\text{Norm} = 2^{19} \cdot 3^3$
$(-2, -2, 3, 1, 0, 0),$	$\text{Norm} = 2^{15} \cdot 3^9$
$(-3, -1, -2, -2, 1, 0),$	$\text{Norm} = 2^{24}$
$(2, -2, -1, -3, 2, 0),$	$\text{Norm} = 2^{28}$
$(-1, 0, -1, -2, 2, 0),$	$\text{Norm} = 2^{16} \cdot 3^6$
$(-3, -1, 0, -2, 3, 0),$	$\text{Norm} = 2^7 \cdot 3^9$
$(3, -3, 2, -2, 3, 0),$	$\text{Norm} = 2^{19} \cdot 3^3$
$(2, -2, 2, -2, 3, 0),$	$\text{Norm} = 2^{13} \cdot 3^3$
$(1, -1, 2, -2, 3, 0),$	$\text{Norm} = 2^{11} \cdot 3^3$
$(-1, 1, 2, -2, 3, 0),$	$\text{Norm} = 2^{23} \cdot 3^3$
$(0, 1, -1, 1, -2, 1),$	$\text{Norm} = 2^{14}$
$(-1, 2, 0, 1, -1, 1),$	$\text{Norm} = 2^{15} \cdot 3^3$
$(3, 2, 2, 1, -1, 1),$	$\text{Norm} = 2^{20}$
$(0, -1, 0, -1, 0, 1),$	$\text{Norm} = 2^6$
$(0, -1, 1, 0, 0, 1),$	$\text{Norm} = 2^{15} \cdot 3^3$
$(2, 0, 2, 0, 0, 1),$	$\text{Norm} = 2^{16}$
$(0, -2, 0, -2, 1, 1),$	$\text{Norm} = 2^{12} \cdot 3^6$
$(-1, 2, -2, 1, -3, 2),$	$\text{Norm} = 2^{24}$
$(0, 0, -3, 1, -2, 2),$	$\text{Norm} = 2^{24}$
$(-1, 0, 0, 0, -1, 2),$	$\text{Norm} = 2^8$
$(0, 0, 0, 0, -1, 2),$	$\text{Norm} = 2^8$
$(1, 0, 0, 0, -1, 2),$	$\text{Norm} = 2^2$
$(2, 0, 0, 0, -1, 2),$	$\text{Norm} = 2^{12}$
$(-3, -2, 1, 0, -1, 2),$	$\text{Norm} = 2^{16}$
$(3, -2, 2, -1, 0, 2),$	$\text{Norm} = 2^{20}$
$(-3, 1, -1, 0, 0, 2),$	$\text{Norm} = 2^9 \cdot 3^9$
$(-3, 1, -2, 1, -3, 3),$	$\text{Norm} = 2^{11} \cdot 3^3$
$(1, 3, -2, 1, -3, 3),$	$\text{Norm} = 2^{21} \cdot 3^3$
$(0, 0, -1, 1, -3, 3),$	$\text{Norm} = 2^{15} \cdot 3^3$
$(-3, 2, -1, 1, -2, 3),$	$\text{Norm} = 2^{13} \cdot 3^3$
$(-1, 1, 3, 0, -1, 3),$	$\text{Norm} = 2^{24}$

From this list, we will select elements with increasingly more complicated norm factorizations.

Next we search for a principal ideal with norm divisible by 2 but not 4. We soon find  $z = (1, 0, -1, 1, 0, -4)$ , with  $\text{Norm}(z) =$

$2 \cdot 246613$ . We arbitrarily label the prime ideal of norm 2 dividing the principal ideal  $(z)$  as  $P_1$ . We label the other five prime ideals as follows:

$$P_2 = \sigma^{-1}(P_1),$$

$$P_3 = \sigma^{-2}(P_1),$$

$$P_4 = \sigma^{-3}(P_1),$$

$$P_5 = \sigma^{-4}(P_1),$$

$$P_6 = \sigma^{-5}(P_1).$$

To start, we set  $r_1 = (1, 0, 0, 0, -1, 2)$ . Then it is easy to see that  $(r_1) = P_2 P_5$  since  $2|_1 z \sigma^{-2}(z) \sigma^{-3}(z) \sigma^{-5}(z)$ . Then we can also find  $r_2 = \sigma(r_1) = (-36, 13, 0, 13, -55, 97)$  with  $(r_2) = P_1 P_4$ , and  $r_3 = \sigma(r_2) = (61, -13, 0, -13, 56, -99)$  with  $(r_3) = P_3 P_6$ .

Next, we set  $r_4 = (0, -1, 0, -1, 0, 1)$ , and compute that  $(r_4) = P_1 P_2^2 P_4 P_5^2$ . Then  $r_4/r_1^2 r_2 = (-221, 80, 0, 80, -338, 596) = u_1$  is a unit in  $O_\Omega$ , and so is its inverse  $u_1^{-1} = (-37, 8, 0, 8, -34, 60)$ , as well as all powers of each. Other units are obtained by taking the images of these under the automorphisms of the Galois group, obtaining  $u_2 = (393, -86, 0, -86, 370, -654)$ ,  $u_2^{-1} = (17, -6, 0, -6, 26, -46)$ ,  $u_3 = (-21, 6, 0, 6, -32, 58)$ , and  $u_3^{-1} = (-5, -2, 0, -2, 8, -14)$ . It turns out that  $U_1 = -u_3^{-1} = (5, 2, 0, 2, -8, 14)$  is the element with the smallest coefficients in the free part of the unit group  $U$ , and the element in  $U \setminus \langle U_1 \rangle$  with the smallest coefficients is  $U_2 = -u_2^{-1} = (-17, 6, 0, 6, -26, 46)$ , and these two units together with  $-1$  generate all of  $U$ .

Next, we set  $r_5 = (0, 0, 0, 0, -1, 2)$ , and compute that  $(r_5) = P_1^3 P_3 P_4 P_6$ . Then

$$(r_5/r_2^3 r_3) U_1^2 U_2^2 = (-1, 0, 0, 0, 0, 0),$$

and we have no further information derived from  $r_5$ .

Next, we set  $r_6 = (-1, 0, 0, 0, -1, 2)$ , and compute that  $(r_6) = P_2^4 P_5^4$ . Then

$$(r_6/r_1^4) U_1^{-3} U_2^{-2} = (-1, 0, 0, 0, 0, 0),$$

and we have no further information derived from  $r_6$ .

Next, we set  $r_7 = (2, 0, 0, 0, -1, 2)$ , and compute that  $(r_7) = P_1 P_3^5 P_4 P_5^5$ . Then

$$(r_7/r_2^5 r_3^5) U_1^3 U_2^{-1} = (-1, 0, 0, 0, 0, 0),$$

and we have no further information derived from  $r_7$ .

Next, we set  $r_8 = (3, 0, 1, 0, 0, 0)$ , and compute that  $(r_8) = P_3^4 P_4^5 P_5^2 P_6$ . Then

$$(r_8/r_3) U_2^{-1} = (-14, 1, -1, 2, -9, 16) = r_9,$$

and  $(r_9) = P_3^3 P_4^5 P_5^2$ .

Next, we set  $r_{10} = (0, 1, -1, 1, -2, 1)$ , and compute that  $(r_{10}) = P_1^3 P_2^5 P_3^4 P_5^2$ . Then

$$(r_{10}/\sigma^2(r_9) r_3^2) U_1^{-2} U_2 = (-1, 0, 0, 0, 0, 0),$$

and we have no further information derived from  $r_{10}$ .

Next, we set  $r_{11} = (-3, -2, 1, 0, -1, 2)$ , and compute that  $(r_{11}) = P_2^{10} P_4^4 P_3 P_6$ . Then

$$(r_{11} r_3^4 / \sigma(r_9)) U_1^{-2} U_2^3 = (4, -2, 5, -3, 2, 5) = r_{12},$$

and  $(r_{12}) = P_2^7 P_4^2 P_5^5$ .

Now, we have enough data to construct the Stickelberger ideal:

$$\begin{aligned}(r) &= (H(\beta)) = 2^9 P_2^5 P_3^3 P_4^{10} P_5^5 P_6^7 \\ &= ([r_9 \sigma^5(r_9) r_{12} / r_1^2] U_1^{-1} U_2^{-1}) \\ &= (512 \cdot (-136, 66, 27, 55, -260, 472)).\end{aligned}$$

We set  $r = 512 \cdot (-136, 66, 27, 55, -260, 472)$ . Since  $r = (-1)^{e_0} U_1^{e_1} U_2^{e_2} H(\beta)$  and  $|r| = |H(\beta)|$ , we can compute that  $e_1 = e_2 = 0$ , and  $H(\beta) = r$  or  $-r$ . Thus, we have found the value of the Gauss sum up to a sign and the choice of  $P_1$ , that is, up to a sign and a power of  $\sigma$ . The twelve possible values of the Gauss sum are the following

$$\begin{aligned}&512 \cdot (-136, 66, 27, 55, -260, 472) \\ &512 \cdot (136, -66, -27, -55, 260, -472) \\ &512 \cdot (101, -53, -10, -38, 173, -312) \\ &512 \cdot (-101, 53, 10, 38, -173, 312) \\ &512 \cdot (-21, -14, -2, -3, 13, -24) \\ &512 \cdot (21, 14, 2, 3, -13, 24) \\ &512 \cdot (35, -41, -27, -30, 155, -287) \\ &512 \cdot (-35, 41, 27, 30, -155, 287) \\ &512 \cdot (13, 18, 10, 3, -23, 47) \\ &512 \cdot (-13, -18, -10, -3, 23, -47) \\ &512 \cdot (-55, 24, 2, 13, -58, 104) \\ &512 \cdot (55, -24, -2, -13, 58, -104).\end{aligned}$$

We will apply the integer coefficient criterion [1] to eliminate some of these possibilities. In essence, it states that the polynomial  $H(x)$  constructed using the Chinese remainder theorem (CRT) from knowledge of  $H(x) \pmod{\Phi_d(x)}$  for  $d \mid N$ , must have rational integer coefficients. The result of applying it in this situation is that only the following cases survive:

$$\begin{aligned}&512 \cdot (136, -66, -27, -55, 260, -472) \\ &512 \cdot (-101, 53, 10, 38, -173, 312) \\ &512 \cdot (21, 14, 2, 3, -13, 24) \\ &512 \cdot (-35, 41, 27, 30, -155, 287) \\ &512 \cdot (-13, -18, -10, -3, 23, -47) \\ &512 \cdot (55, -24, -2, -13, 58, -104).\end{aligned}$$

The last case yields

$$\begin{aligned}H(x) &= 1305 - 231x - 231x^2 + 281x^3 - 231x^4 + 281x^5 \\ &\quad + 281x^6 + 1305x^7 - 231x^8 + 1305x^9 + 281x^{10} \\ &\quad - 231x^{11} + 281x^{12} - 1767x^{13} + 1305x^{14} \\ &\quad + 1305x^{15} - 231x^{16} - 1767x^{17} + 1305x^{18} \\ &\quad - 231x^{19} + 281x^{20} - 231x^{21} - 231x^{22} - 231x^{23} \\ &\quad + 281x^{24} + 1305x^{25} - 1767x^{26} - 231x^{27} + 1305x^{28} \\ &\quad - 1767x^{29} + 1305x^{30} + 281x^{31} - 231x^{32} + 281x^{33} \\ &\quad - 1767x^{34} - 743x^{35} + 1305x^{36} + 281x^{37} - 231x^{38} \\ &\quad + 793x^{39} + 281x^{40} - 231x^{41} - 231x^{42} - 4327x^{43} \\ &\quad - 231x^{44} - 743x^{45} - \dots + 793x^{214}.\end{aligned}$$

This can be represented in shorthand notation as

$$(0, 1035)(1, -231)(3, 281)(5, 281)(7, 1305)(13, -1767) \\ (15, 1305)(19, -231)(35, -743)(39, 793)(43, -4327),$$

TABLE I  
CLASSES OF CODEWORDS

$i$	Size	$\eta_i$	$W_i$
0	1	1305	623616
1	28	-231	624384
3	28	281	624128
5	14	281	624128
7	28	1305	623616
13	28	-1767	625152
15	14	1305	623616
19	28	-231	624384
35	14	-743	624640
39	28	793	623872
43	4	-4327	626432

From this, the weight enumerator is computed to be

$$\begin{aligned}w(z) &= 1 + 1248537 \\ &\quad \cdot (43z^{623616} + 28z^{623872} + 42z^{624128} \\ &\quad + 56z^{624384} + 14z^{624640} + 28z^{625152} + 4z^{626432}).\end{aligned}$$

The Appendix gives these shorthand forms of the generating functions  $H(x)$  for all codes with  $p = 2$ ,  $k > 27$ , and  $N < 500$ . The cases with  $k \leq 27$  have previously been tabulated [2].

APPENDIX  
SHORTHAND FORMS OF  $H(x)$

$N$	$k$	$(i, \eta_i)$
29	28	(0, 15819)(1, -565)
37	36	(0, 255059)(1, -7085)
53	52	(0, 65842659)(1, -1266205)
69	58	(0, 527771405)(1, -9099507)
61	60	(0, 1056139499)(1, -17602325)
67	66	(0, -128207979)(1, 8461726613)
71	35	(0, 169609)(1, -10615)(7, 5769)
77	30	(0, 27515)(1, -133)(3, 891)(7, -5253)(11, -133) \\ (33, 891)
79	39	(0, 452945)(1, 59729)(3, -71343)
81	54	(0, 132560719)(1, -1657009)(3, -1657009)(9, -1657009) \\ (27, -1657009)
83	82	(0, 2172528999461)(1, -26494256091)
87	28	(0, 13465)(1, 1177)(3, -871)(5, -871) \\ (29, 1177)
95	36	(0, 42081)(1, -23455)(5, 9313)(7, 25697) \\ (19, -72607)
97	48	(0, 16604255)(1, -172961)(5, -172961)
99	30	(0, 32437)(1, -331)(3, -331)(5, -331)(9, -331) \\ (11, -331)(15, -331)(33, -331)
101	100	(0, 1114752383012499)(1, -11147523830125)
103	51	(0, 20767913)(1, -4397911)(3, 3990697)
107	106	(0, 8923019822453693)(1, -84179432287299)
109	36	(0, 259739)(1, -2405)(3, -2405)(9, -2405)
111	36	(0, 106865)(1, 24945)(3, -7823)(11, -24207) \\ (37, 74097)
113	28	(0, 16239)(1, -145)(3, -145)(5, -145)(9, -145)
115	44	(0, -763067)(1, 219973)(5, 678725)(7, -369851) \\ (23, 809797)(25, -304315)
121	110	(0, 35731038365914679)(1, -297758653049289)

- (9, 74953)(15, 37577)(21, 9417)(27, -56119)  
(45, -27959)(63, -56119)
- 137 68 (0, 17054468679)(1, -125400505)(3, -125400505)
- 139 138 (0, 586049077910081870045)(1, -4246732448623781667)
- 141 46 (0, 4865467)(1, -377413)(3, -377413)(5, 409019)  
(15, 409019)(47, -3523141)
- 143 60 (0, 7611924849)(1, -44113519)(5, 56549777)  
(11, -144776815)(13, 22995345)
- 145 28 (0, 16271)(1, -113)(3, -113)(5, -113)(7, -113)  
(11, -113)(29, -113)
- 147 42 (0, -449947)(1, -23963)(3, 139877)(5, -23963)  
(7, -105883)(9, -73115)(21, 598629)(35, -105883)  
(49, 942693)(63, -449947)
- 149 148 (0, 18762690992341140714819)  
(1, -126774939137440139965)
- 157 52 (0, 66681419)(1, -427445)(3, -427445)(9, -427445)
- 159 52 (0, 13558945)(1, -5315423)(3, -1121119)  
(11, 5170337)(53, 26141857)
- 161 33 (0, 50335)(1, 3231)(3, 1183)(5, -2913)(7, 9375)  
(11, -865)(23, 7327)(35, -19297)(69, 5279)
- 163 162 (0, 2403018193589815046655221)  
(1, -14833445639443302757131)
- 167 83 (0, -102461974295)(1, -239900927767)  
(5, 241135409385)
- 169 156 (0, 300443103099493641051239)  
(1, -1788351804163652625305)  
(13, -1788351804163652625305)
- 173 172 (0, 76924019782184034422922459)  
(1, -447232673152232758272805)
- 175 60 (0, -295392847)(1, -2840143)(3, 8694193)  
(5, -58414671)(7, -60511823)(15, -36394575)  
(25, -36394575)(35, 241478065)(75, 478456241)
- 177 58 (0, 533837743)(1, -3033169)(3, -3033169)  
(5, -3033169)(59, -3033169)
- 179 178 (0, 615512086572060583609061765)  
(1, -3457933070629553840500347)
- 181 180 (0, 1231100591554521820341670499)  
(1, -6839447730858454557453725)
- 183 60 (0, 978395897)(1, -28237063)(3, -28237063)  
(5, 38871801)(61, 38871801)
- 185 36 (0, 260727)(1, -1417)(3, -1417)(5, -1417)  
(9, -1417)(19, -1417)(37, -1417)
- 187 40 (0, -148595)(1, -83059)(3, -17523)(9, 48013)  
(11, -214131)(17, 113549)(23, 48013)(33, 113549)
- 191 95 (0, -167344518426431)(1, 8577342017729)  
(7, -6815820771135)
- 193 96 (0, 280016557142207)(1, -1458419568449)  
(5, -1458419568449)
- 197 196 (0, 315303956401945384128621609459)  
(1, -160869365511966245554191885)
- 199 99 (0, -752662673277175)(1, 21393512677129)  
(3, -13790859411703)
- 201 66 (0, 8547198599)(1, -42735993)(3, -42735993)  
(7, -42735993)(67, -42735993)
- 203 84 (0, 4346354834205)(1, 19980589853)(3, -53302289635)  
(7, -51691676899)(29, 1980589853)(87, -53302289635)
- 207 66 (0, 6743062993)(1, 82508241)(3, 82508241)  
(5, -1377839)(9, 82508241)(15, -1377839)  
(23, -1846871599)(45, -1377839)(69, -1846871599)
- 209 90 (0, 35016025810895)(1, -168346277937)  
(3, -168346277937)(11, -168346277937)  
(19, -168346277937)
- 211 210 (0, 40372568879306642550037182654629)  
(1, -192250327996698297857319917403)
- 213 70 (0, 14746306435)(1, -822950013)(3, -822950013)  
(7, 1056098179)(21, 1056098179)(71, -19613431933)
- 215 28 (0, 1305)(1, -231)(3, 281)(5, 281)(7, 1305)  
(13, -1767)(15, 1305)(19, -231)(35, -743)  
(39, 793)(43, -4327)
- 223 37 (0, 25825)(1, -15135)(3, -31519)(5, 9441)  
(9, 25825)(13, 34017)(19, -23327)
- 225 60 (0, 220511455)(1, 17611999)(3, 24657119)  
(5, 39173343)(7, -11846433)(9, -76006177)  
(15, -249250593)(21, -889731320)(25, 173391071)  
(35, -95044385)(45, 86293727)(75, -115032865)  
(105, 287620319)
- 227 226 (0, 10338846608183885850642221765671733)  
(1, -45747108885769406418770892768459)
- 229 76 (0, 273677566739)(1, -1200340205)  
(3, -1200340205)(5, -1200340205)
- 231 30 (0, 8489)(1, 2345)(3, 297)(5, 2345)  
(7, -2775)(9, -727)(11, -1751)(17, -1751)  
(21, 297)(23, -1751)(33, 4393)(35, -2775)  
(55, -1751)(77, 9513)(99, 3369)
- 233 29 (0, -9049)(1, 2215)(3, 2215)(5, 167)  
(7, -857)(9, -1881)(17, -857)(27, 167)(29, -857)
- 235 92 (0, 27663537100861)(1, 450624313405)  
(5, 9590314719293)(11, -1404801558467)  
(25, -8345468708803)(47, 7872327800893)
- 237 78 (0, -37766716645)(1, -3406978277)  
(3, 65312498459)(7, -3406978277)  
(9, -54946585829)(79, 82492367643)
- 239 119 (0, -558352136228515343)(1, 40626614211760625)  
(7, -35934579453537807)
- 243 162 (0, 2407901632483458932336581)  
(1, -9950006745799417075771)  
(3, -9950006745799417075771)  
(9, -9950006745799417075771)  
(27, -9950006745799417075771)  
(81, -9950006745799417075771)
- 245 84 (0, -2347877837661)(1, 37171188899)  
(3, -112884231005)(5, 230713152675)  
(7, 572699923619)(15, -65908026205)  
(21, -526811704157)(35, 950657045667)  
(49, 572699923619)(105, -148854582109)
- 247 36 (0, -49351)(1, -16583)(3, -16583)(5, 16185)  
(11, 16185)(13, 16185)(15, 16185)(17, -16583)  
(19, -16583)
- 249 82 (0, 2190191836855)(1, -8831418697)  
(3, -8831418697)(11, -8831418697)(83, -8831418697)
- 251 50 (0, 33420749)(1, -133683)(3, -133683)  
(7, -133683)(9, -133683)(17, -133683)
- 253 110 (0, -9872845193878613)(1, 736342502532011)  
(5, -1364824218147925)(11, 7642375036593067)  
(23, 3637953688232875)(55, -3767257124838485)
- 259 36 (0, 97877)(1, 11861)(3, -25003)(5, 7765)  
(7, -427)(11, 11861)(13, 7765)(23, -20907)  
(37, 44629)(111, 7765)
- 261 84 (0, 2361606691891)(1, 145403567155)  
(3, -223963620301)(5, -155244143565)  
(9, -223963620301)(15, 325792193587)  
(29, 626439904307)(87, -936928191437)
- 263 131 (0, 40485956485918384969)  
(1, 1863086081589011273)(5, -2172139184534953143)
- 265 52 (0, 66855623)(1, -253241)(3, -253241)  
(5, -253241)(7, -253241)(19, -253241)(53, -253241)

- 269 268 (0, 21697112109397533551389595786876164917819)  
(1, -80959373542528110266379088757000615365)
- 271 135 (0, -204462758344668138991)  
(1, 3063112484564316689)(3, -1548573533863071215)
- 277 92 (0, 70114705390019)(1, -254038787645)  
(3, -254038787645)(5, -254038787645)
- 279 30 (0, -1383)(1, -487)(3, 857)(5, -487)  
(7, -39)(9, -4583)(11, -1831)(15, -1639)  
(17, 345)(21, -2855)(23, 1945)(27, -1639)  
(31, -871)(33, 537)(45, 5593)(51, 3609)  
(63, 537)(69, -2599)(93, 6809)(99, 857)  
(135, 5337)
- 281 70 (0, 34237461719)(1, -122276649)  
(3, -122276649)(5, -122276649)(15, -122276649)
- 283 94 (0, 140240182742765)(1, -497305612563)  
(3, -497305612563)(5, -497305612563)
- 285 36 (0, -147765)(1, -8501)(3, -11061)  
(5, -3381)(7, 12491)(11, -3893)(15, 16075)  
(17, 7883)(19, -24885)(21, 5323)(25, -3381)  
(57, -27445)(95, 94923)(133, -20277)
- 287 60 (0, 346932513)(1, 18728225)(3, 13485345)  
(5, 13485345)(7, 78497057)(11, -48380639)  
(21, -122829535)(41, -120732383)(123, 354272545)
- 289 136 (0, 294126632151050566943)  
(1, -1021273028302258913)(3, -1021273028302258913)  
(17, -1021273028302258913)(51, -1021273028302258913)
- 291 48 (0, 2738549)(1, 1165685)(3, -931467)  
(5, -931467)(13, 117109)(15, 641397)  
(25, -407179)(97, 6932853)
- 293 292 (0, 88898533760695453342367641441820089552850259)  
(1, -304447033427039223775231648773356471071405)
- 295 116 (0, -244018767606406807)(1, -9831586983141015)  
(3, 8182811526340969)(5, 8182811526340969)  
(59, -9831586983141015)
- 297 90 (0, 35065906189543)(1, -118465899289)  
(3, -118465899289)(5, -118465899289)  
(9, -118465899289)(11, -118465899289)  
(15, -118465899289)(27, -118465899289)  
(33, -118465899289)(45, -118465899289)  
(99, -118465899289)
- 299 132 (0, 63164209519505656445)(1, 712512011782277757)  
(5, -262798782520145283)(13, 712512011782277757)  
(23, -10622766775332550019)(65, -262798782520145283)
- 301 42 (0, -177829)(1, -30373)(3, 51547)  
(7, -308901)(9, 35163)(13, 51547)(21, 280923)  
(23, -95909)(27, -79525)(43, 428379)(49, 84315)  
(129, 313691)
- 303 100 (0, 793796592736817)(1, 54924778871345)  
(3, -50628337395151)(5, 54924778871345)  
(101, 160477895137841)
- 305 60 (0, 1070221359)(1, -3520465)(3, -3520465)  
(5, -3520465)(9, -3520465)(13, -3520465)  
(61, -3520465)
- 307 102 (0, 2244464960871941)(1, -7334852813307)  
(5, -7334852813307)(7, -7334852813307)
- 309 102 (0, -438381335087389)(1, -16168870021405)  
(3, -156906358376733)(5, -16168870021405)  
(9, 194937362511587)(103, 898624804288227)
- 311 155 (0, 197260346723647529319289)  
(1, 3938468831171428383609)  
(11, -5211116229388509217927)
- 313 156 (0, 301265859201089698489079)  
(1, -965595702567595187465)
- (5, -965595702567595187465)
- 317 316 (0, 364222805517795995388300  
005629609963889017889259)  
(1, -11526038149297341626212025  
49460791024965246485)
- 319 140 (0, -500086027427712233151)  
(1, -57364169658682994367)(3, 53316294783574315329)  
(11, -20470681511263891135)(29, 16399675922  
5831625025)
- 321 106 (0, 8979139443978559)(1, -28059810762433)  
(3, -28059810762433)(7, -28059810762433)  
(107, -28059810762433)
- 323 72 (0, -21009437547)(1, -3829568363)(3, 2612882581)  
(5, -1682084715)(9, 2612882581)(17, 4760366229)  
(19, 6907849877)(57, -12419502955)
- 325 60 (0, 1070438003)(1, -3303821)(3, -3303281)  
(5, -3303821)(7, -3303821)(11, -3303821)  
(13, -3303821)(15, -3303821)(25, -3303821)  
(35, -3303821)(55, -3303821)(65, -3303821)
- 327 36 (0, 206729)(1, 1929)(3, -6263)(5, -14455)  
(7, 18313)(9, 1929)(11, 1929)(19, 1929)  
(27, -6263)(35, -6263)(109, 26505)
- 329 69 (0, 8870379271)(1, -633913593)(3, -440975609)  
(5, 498548487)(7, -256426233)(11, 305610503)  
(35, 683097863)(47, -8494039289)(141, 8492891911)
- 331 30 (0, 32669)(1, -99)(3, -99)(5, -99)(7, -99)  
(9, -99)(15, -99)(17, -99)(19, -99)(23, -99)  
(25, -99)(29, -99)
- 333 36 (0, 57467)(1, 8315)(3, 8315)(7, 24699)  
(9, -8069)(11, -24453)(13, 8315)(29, -8069)  
(31, -8069)(33, -8069)(37, 24699)(111, 24699)
- 335 132 (0, 10250676720810429009)  
(1, -3584381334471734703)(5, 739074307803941457)  
(11, 3765493257396914769)(67, -20734088715498583471)
- 339 28 (0, 6437)(1, 293)(3, 293)(5, -475)(9, 1061)  
(11, -475)(15, -1243)(17, 37)(19, 1061)(23, -475)  
(27, -731)(31, -731)(83, 805)(113, 4901)
- 343 147 (0, -2895538122429535910503)  
(1, 7358911333686543769)(3, -78208918636399458919)  
(7, -534354880994713303655)  
(21, 646236739722697999769)  
(49, 6549194843309754516889)  
(147, -2895538122429535910503)
- 345 44 (0, -2987753)(1, 125207)(3, 57623)(5, 194839)  
(7, -159465)(13, 37143)(15, 289047)(17, -71401)  
(21, -138985)(23, 321815)(25, -132841)  
(69, 254231)(75, -38633)(115, 1112343)(161, 233751)
- 347 346 (0, 11938118181276962018011258812004  
840092130847808796973)  
(1, -345032317377946879133273376069  
50404890551583262419)
- 349 348 (0, 238766318150666781298667964473633  
41506953850936599819)  
(1, -68611010962835281982375851860239  
487088947847518965)
- 351 36 (0, -41631)(1, -7327)(3, 24929)(5, 5985)  
(7, -3743)(9, 15201)(13, 17761)(15, -40607)(17, 2913)  
(21, 33633)(25, 5473)(27, -17567)(29, -7327)  
(39, -13983)(51, 865)(63, -17567)(75, -7839)  
(87, 865)(117, 23905)
- 353 88 (0, 17542349823327)(1, -49836221089)  
(3, -49836221089)(7, -49836221089)(9, -49836221089)
- 355 140 (0, 934122375424734295989)(1, -27846504981603649611)

- (5, -27846504981603649611)(7, 28142245585866356661)  
(35, 28142245585866356661)(71, -2464692452926  
77007435)
- 359 179 (0, 258592263548821696253097385)  
(1, 43403467657417703155398057)  
(7, -44848117174450226598152791)
- 361 342 (0, 2984864064186504441920256934806263558814199  
294419239)  
(1, -829128906718473456088960259668406544115055  
3595609)  
(19, -829128906718473456088960259668406544115055  
3595609)
- 363 110 (0, 35929544134614205)(1, -99252884349763)  
(3, -99252884349763)(5, -99252884349763)  
(11, -99252884349763)(33, -99252884349763)  
(55, -99252884349763)(121, -99252884349763)
- 365 36 (0, -41061)(1, 8091)(3, 16283)(5, 20379)  
(7, -12389)(9, -8293)(11, -16485)(13, 8091)  
(15, 49051)(17, -101)(25, -101)(27, -101)  
(45, -101)(55, -24677)(65, -8293)(73, -8293)  
(85, -16485)(125, 8091)
- 367 183 (0, -3163577695762332494062714255)  
(1, 86014907361790727547486833)  
(3, -68727597548881806814903695)
- 369 60 (0, -40542097)(1, 31285359)(3, 65626223)  
(7, 1138799)(9, -44212113)(11, 13983855)  
(21, -44212113)(23, -40542097)(27, -1482641)  
(33, 90005615)(41, 60645487)(69, -135700369)  
(123, 362111087)
- 371 156 (0, 291649485839580994925637)  
(1, 939189042258805191749)(3, 687307218699600720965)  
(7, -10581969064076298750907)(53, 93918904  
2258805191749)  
(159, 687307218699600720965)
- 373 372 (0, 9781676631886080949006843437639292504985238  
8796105095459)  
(1, -26294829655607744486577536122686270174691  
5023645443805)
- 375 100 (0, 631691447949241)(1, 11723119318969)  
(3, -9210249291847)(5, -3689974802503)  
(7, -11520069281863)(15, -36811621738567)  
(25, -320349323601991)(35, 31494397286329)  
(75, 68741494527929)(125, 242600629819321)  
(175, 242600629819321)
- 377 84 (0, 4386380605239)(1, -11665905865)  
(3, -11665905865)(9, -11665905865)(13, -11665905865)  
(17, -11665905865)(29, -11665905865)
- 379 378 (0, 7825674327098170609267573885133568289  
20412387205608524365)  
(1, -207028421351803455271628938760147309  
2382043352395789747)
- 383 191 (0, 54867328745102456498556125569)  
(1, 397967016545724402994659713)  
(5, -68523051782883337542073983)
- 385 60 (0, -701644417)(1, -4341377)(3, 17678719)  
(5, 901503)(7, -30555777)(11, -54673025)  
(13, 901503)(15, -37895809)(21, -13778561)  
(23, 12435839)(33, -49430145)(35, 137216383)  
(55, 235782527)(77, -80887425)(165, 196985215)
- 387 42 (0, 2091733)(1, -5419)(3, -5419)(5, -5419)  
(7, -5419)(9, -5419)(11, -5419)(13, -5419)  
(15, -5419)(19, -5419)(21, -5419)(27, -5419)  
(33, -5419)(39, -5419)(43, -5419)(57, -5419)
- (63, -5419)(129, -5419)
- 389 388 (0, 2504386090827796541252736551367171793776569  
5757861649264819)(1, -645460332687576428157  
92179158947726643726019994488786765)
- 391 88 (0, -3127472157751)(1, -90930279479)  
(3, -640686093367)(5, 780948081609)  
(15, -318563546167)(17, -318563546167)  
(23, -2577716343863)(69, 3469597608905)  
(85, 2108092976073)
- 393 130 (0, 36799611587247553351)(1, -93876560171549881)  
(3, -93876560171549881)(5, -93876560171549881)  
(131, -93876560171549881)
- 395 156 (0, 294909545225549753573853)(1, 24179711928111  
03150557)(3, -3780134815955306192419)  
(5, 2417971192811103150557)  
(15, -3780134815955306192419)(79, -7321909678  
107540102691)
- 397 44 (0, 4183739)(1, -10565)(3, -10565)(5, -10565)  
(7, -10565)(9, -10565)(15, -10565)(21, -10565)  
(23, -10565)(27, -10565)
- 401 200 (0, 1264489376786263742141349830799)  
(1, -3161223441965659355353374577)  
(3, -3161223441965659355353374577)
- 403 60 (0, -247837851)(1, -37835931)(3, -7783579)  
(5, 16493413)(7, 22174565)(11, -20178075)  
(13, 248257381)(15, 6490981)  
(31, 20597605)(39, 230599525)  
(65, -7783579)(91, -261944475)(143, 22174565)  
(195, 16493413)
- 405 108 (0, -2548827592081853)(1, -170667693064637)  
(3, -186802408800701)(5, 313750125526595)  
(7, 124992621100611)(9, -1558992920265149)  
(15, -367396522574269)(21, 481700660887107)  
(27, -3810792733950397)(45, 1954772035288643)  
(63, 692806893420099)(81, -3810792733950397)  
(135, -2548827592081853)(189, 5196406520790595)
- 407 180 (0, 962340596417754345302225369)  
(1, -33814278944700094655663655)  
(5, 38721270232177655826706905)  
(11, -57992795336992678149787175)  
(37, 24214160396802105730232793)
- 409 204 (0, 5058204840030489934578532105559)  
(1, -12397560882427671408280715945)  
(7, -12397560882427671408280715945)
- 411 68 (0, 316113773)(1, -757628051)(3, 47678317)  
(5, 584549229)(7, 852984685)(9, -220757139)  
(23, -757628051)(137, 8369177453)
- 413 174 (0, -77213648908473977259789493)  
(1, -268288827473617020357813)  
(3, -658497128320437441367221)  
(7, 157603546862289921405771)  
(59, -658497128320437441367221)  
(177, 77102963627862650160837451)
- 415 164 (0, -2043521596288343050738271)  
(1, -230132866866399288679007)  
(3, 223214315489086651835809)(5, 720985880372  
58004997537)(83, -683480049221885229193823)
- 417 138 (0, 588880232875831057823)(1, -1415577482874593889)  
(3, -1415577482874593889)(5, -1415577482874593889)  
(139, -1415577482874593889)
- 419 418 (0, 8207886940365647448780085717061158840  
4276362571403798185821685)  
(1, -1963609256946546589683734108063664087

- 091570245386133488482827)
- 421 420 (0, 164159599542733144348001676169616914986  
4012027472930029943980499)  
(1, -3908561893874598674952420861181355118  
723838160649833404628525)
- 423 138 (0, 571048809636316015081)(1, 5612869220695501289)  
(3, 5612869220695501289)(5, -7627713683773756951)  
(9, 5612869220695501289)(15, -7627713683773756951)  
(45, -7627713683773756951)(47, -192470007223  
89636631)(141, -19247000722389636631)
- 425 40 (0, 52583)(1, -2713)(3, -13977)(5, -71321)  
(7, -16025)(9, 16743)(13, 24935)(15, 201063)(17, 36199)  
(21, -42649)(25, -61081)(29, 26983)(35, -61081)  
(37, 9575)(45, -71321)(65, 190823)(75, -71321)  
(85, 52583)(105, -71321)(145, -61081)(185, -61081)
- 427 60 (0, 321195773)(1, -2814211)(3, -20640003)  
(5, -20640003)(7, -14348547)(9, 64294653)  
(11, -69923075)(33, 46468861)(61, 314904317)  
(183, -69923075)
- 429 60 (0, -415959589)(1, -27986469)(3, -18549285)  
(5, 5567963)(7, 2422235)(11, -61540901)  
(13, -15403557)(15, 15005147)(17, 35976667)  
(33, -52103717)(39, 53802459)(65, -15403557)  
(77, -31132197)(143, 588576219)
- 431 43 (0, -120143)(1, -38223)(5, 51889)(7, -267599)  
(11, 101041)(13, 101041)(19, -111951)(23, -21839)  
(31, 289457)(35, 2737)(47, -103759)
- 433 72 (0, 68560771247)(1, -158705489)(3, -158705489)  
(5, -158705489)(7, -158705489)(9, -158705489)  
(21, -158705489)
- 435 28 (0, 4229)(1, 645)(3, 773)(5, -123)(7, -507)  
(9, -763)(11, 1029)(15, -379)(19, 645)(21, -251)  
(23, -891)(25, 389)(29, -2043)(33, -251)(37, -507)  
(47, -379)(73, 5)(87, 2309)(145, 6021)(203, -379)
- 437 198 (0, -250825011030991080830674904221)  
(1, 9473307605492901547200502627)  
(5, -11311371499456305412260471965)  
(19, 147144953529072369774827428707)  
(23, -13140523488198068050043053213)  
(95, -697548549087714360463447709)
- 439 73 (0, 28924990457)(1, -1139780615)(3, 3155186681)  
(5, 3155186681)(11, -7582231559)(15, 5302670329)  
(23, -3287264263)
- 441 42 (0, 631)(1, 58487)(3, 17143)(5, -3209)  
(7, -210057)(9, -17033)(11, -38921)(13, -28425)  
(15, -28041)(21, 298999)(23, -43529)  
(27, 105591)(31, 7671)(35, 314231)  
(49, 314231)(63, 631)(77, 314231)(91, -210057)  
(105, -225289)(147, -225289)(161, -210057)  
(189, 631) (217, -210057)
- 443 442 (0, 3362386124966304398770547592681342200  
26738652224303904245992023181)(1, -760720842752557  
5562829293196111633937256530593267655891959257971)
- 445 44 (0, -1159573)(1, 3691)(3, 85611)(5, -430485)  
(9, 32363)(11, -135573)(13, -78229)(15, -602517)  
(17, 93803)(19, -33173)(25, 388715)(33, -82325)  
(45, 3691)(55, 556651)(65, 441963)(89, 413291)  
(95, 85611)(165, -33173)
- 447 148 (0, 941568037552029372865)(1, -8293193935  
24087582271)(3, -91449630575705517631)(5, 7939940  
8496235295937)(149, 8910561477394555670977)
- 449 224 (0, 5180732722992433803077199010001343)  
(1, -11564135542393825453297319218753)
- (3, -11564135542393825453297319218753)
- 453 30 (0, 755)(1, -1293)(3, -269)(5, 1779)(7, 1779)  
(9, -2317)(11, -1293)(13, -269)(15, -2317)(17, 1779)  
(21, -269)(23, -2317)(33, 3827)(35, -269)(37, -269)  
(45, -1293)(51, -269)(65, -269)(69, 1779)(105, 755)  
(111, 1779)(151, -1293)
- 457 76 (0, 274276423559)(1, -601483385)(3, -601483385)  
(5, -601483385)(7, -601483385)(13, -601483385)  
(31, -601483385)
- 459 72 (0, 4386579485)(1, -400956387)(3, -3562675171)  
(5, 1538384925)(9, 6553723933)(11, -810949603)  
(15, 1185014813)(17, 2831541277)(19, -398072803)  
(27, 1806033949)(33, -962468835)(45, 1806033949)  
(51, 91612189)(57, 3953517597)(81, -10626145251)  
(99, -15373835235)(153, 4386579485)(171, 6553723933)
- 461 460 (0, 17216937741451514871041148029541386703  
51144320606495561483059188149499)(1, -374281255  
248945975457416261511769276163292243610  
1077307571867800325)
- 463 231 (0, 1813378838067130944996503735958737)  
(1, 2556897805066826564373548948298449)  
(3, -2635399053468001064156947380921647)
- 467 466 (0, 13773934893380739721390133755122457648146538676  
794004859553474468053413)(1, -295578002003878534793  
77969431593256755679267546768250771573979545179)
- 469 66 (0, -407187325)(1, 49991811)(3, 87740547)  
(5, 221958275)(7, -4534141)(9, 452644995)  
(19, -583348093)(25, -352661373)(67, 49991811)  
(201, 2906312835)
- 471 52 (0, 32013849)(1, 3702297)(3, 2653721)(5, 556569)  
(7, -1540583)(9, -1540583)(11, -3637735)  
(13, 2653721)(27, -2589159)(43, -1540583)  
(157, 17333785)
- 473 70 (0, 34287096215)(1, -72642153)(3, -72642153)  
(7, -72642153)(11, -72642153)(13, -72642153)  
(19, -72642153)(21, -72642153)(33, -72642153)  
(43, -72642153)(77, -72642153)
- 475 180 (0, 559547547657268992258993581)  
(1, 14902854088464200827792813)  
(5, -119805011018633863715492435)  
(7, -10381876652440597057638995)  
(19, -90206398495627644927349331)  
(25, 34937493892038670646898093)  
(35, 112308746347374937828093357)  
(95, 17948780469915121990626733)
- 477 156 (0, 26540494738671128256907)(1, 72544238096077920  
42379)(3, -24566209717541184495221)(9, -56767  
43786062603640437)(11, -6626751105858645548661)  
(33, 22657455111155267641739)(53, 172617824  
69595223794059)(159, 83208892533106870821259)
- 479 239 (0, -33760430702776391306252273986107935)  
(1, 42988207237441529577964124880176609)  
(13, -42846950205212339656180642562243103)
- 481 36 (0, 261599)(1, -545)(3, -545)(5, -545)(7, -545)  
(9, -545)(11, -545)(13, -545)(17, -545)  
(19, -545)(21, -545)(25, -545)(37, -545)(41, -545)  
(51, -545)
- 483 66 (0, 6168032309)(1, -47926219)(3, -140200907)  
(5, 69514293)(7, -421219275)(9, 140817461)  
(11, 69514293)(13, -47926219)(15, 258257973)  
(21, -710626251)(23, -568019915)(33, -22760395)  
(35, 535082037)(69, -660294603)(105, 245675061)  
(115, -568019915)(161, -2132495307)(207, -379276235)



- 485 48 (0, 5942291)(1, -349165)(5, 207891)(7, -742381)  
(11, -611309)(13, 1092627)(17, -381933)(25, -611309)  
(29, -349165)(47, 1256467)(53, 142355)(97, 2665491)
- 487 243 (0, 829061609280632846806019290677935657)  
(1, 164447611388174910354115760537763369)  
(3, -167859387558054057871836004532322775)
- 489 162 (0, 2412907157349443915159975)(1, -49444818798  
14434252377)(3, -4944481879814434252377)  
(7, -4944481879814434252377)(163, -4944481  
879814434252377)
- 491 490 (0, 56423955144042904863271089229427621849643477  
602871090149382454656350871869)(1, -11515092886  
5393683394430794345770656836007097148716510508  
943785012960963)
- 493 56 (0, -80380901)(1, 11893787)(3, 11893787)  
(5, -21660645)(9, -689125)(15, -13272037)  
(17, 11893787)(23, -689125)(25, 3505179)  
(29, 3505179)(61, 7699483)(87, -25854949)
- 495 60 (0, 93369073)(1, 6861553)(3, 44380913)(5, 17806065)  
(7, -25021711)(9, -29019407)(11, 44184305)  
(15, 114340593)(19, 21541617)(21, -56282383)  
(25, -11554063)(27, 10826481)(29, 4338417)  
(33, 46478065)(45, 47231729)(55, 30388977)  
(57, 4535025)(75, -154094863)(77, -53955855)  
(87, 38089457)(99, -289066255)(165, 160477937)  
(231, -121294095)
- 497 105 (0, 5372184431243503)(1, 110299737829615)  
(3, 177885343199471)(7, 72538385363183)  
(11, -166777192369937)(13, -99191587000081)  
(49, -213334637858577)(71, -496630680702737)  
(213, -429045075332881)
- 499 166 (0, 9652024980650666597253317)  
(1, -19381576266366800396091)  
(5, -19381576266366800396091)  
(11, -19381576266366800396091)

## REFERENCES

- [1] R. Segal and R. Ward, "Weight distributions of some irreducible cyclic codes," *Math. Comput.*, vol. 46, no. 173, 341-354, Jan. 1986.
- [2] F. MacWilliams and J. Seery, "The weight distributions of some minimal cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-27, no. 6, pp. 796-806, Nov. 1981.

## On the Generalized Hamming Weights of Product Codes

Victor K. Wei, *Member, IEEE*, and Kyeongcheol Yang

**Abstract**—The  $r$ th generalized Hamming weight of a linear code is the minimum support size of any  $r$ -dimensional subcode. It has been found useful in the studies of cryptography and trellis coding. We derive several results on expressing the generalized Hamming weights of a product code in terms of those of its component codes. We also formulate a general conjecture.

**Index Terms**—Generalized Hamming weights, product code.

## I. INTRODUCTION

V. K. Wei introduced the notion of generalized Hamming weights [11] for linear codes. The  $r$ th generalized Hamming weight is the minimum support size of any  $r$ -dimensional subcode. These parameters have been found useful in the studies of cryptography, including the wire-tap channel of type II, and  $t$ -resilient functions [11].

Many interesting results have been obtained on this topic. Wei [11] derived some basic properties of these weights and he determined the generalized weights for several classes of codes, notably Reed-Muller codes of all orders. Feng, Tzeng, and Wei [3] obtained bounds on the generalized Hamming weights of several classes of BCH codes and other cyclic codes. Chung [2] determined the second weight of the dual of the double-error-correcting BCH code for a majority of the cases. Yang, Kumar, and Stichtenoth [13] studied the generalized weights of algebraic geometric codes. Kløve, Hellese, and Ytrehus [5] derived general bounds on these parameters. Kløve [7], [8] obtained relationships among the first few weights.

Recently, another application of generalized Hamming weights emerged in the trellis decoding of linear block codes. In order to exploit the advantage of maximum-likelihood soft-decision decoding, several researchers [1], [4], [6], [12] studied the decoding of block codes by Viterbi algorithm, i.e. dynamic programming. The dynamic programming steps can be specified by a trellis diagram. One central issue is the minimum number of trellis states required to perform successive dynamic programming stages. The base-two logarithm of this number is called the minimum trellis size of the linear block code.

In general, the minimum trellis size may vary with the ordering of the bits. However, Kasami *et al.* [6], derived a necessary and sufficient condition for the existence of an "optimal bit ordering". This condition was expressed partly in terms of the generalized Hamming weights of the code. Furthermore, using results from [11], Kasami *et al.* 1) showed that any Reed-Muller code of arbitrary order has an optimal bit ordering, and 2) determined the minimum trellis size of these codes.

The main goal of this study is to express the generalized Hamming weights of a product code in terms of those of its component codes. We are successful here for several cases, notably the product of two parity-check codes and the product of a parity-check code and a dual

Manuscript received December 4, 1991; revised September 10, 1992. The work of K. Yang was supported in part by Bellcore summer research program and in part by NSF under Grant NCR-9016077. The work of K. Yang was done while the author was with the University of Southern California.

V. K. Wei is with Bellcore, Morristown, NJ 07960.

K. Yang is with the Department of Electronic Communication Engineering, Hanyang University Seoul, Korea.

IEEE Log Number 9210706.