517-61 N93-19430
128450
p. 8

Binary Weight Distributions of Some Reed-Solomon Codes

F. Pollara and S. Arnold
Communications Systems Research Section

The binary weight distributions of the (7,5) and (15,9) Reed-Solomon (RS) codes and their duals are computed using the MacWilliams identities. Several mappings of symbols to bits are considered and those offering the largest binary minimum distance are found. These results are then used to compute bounds on the soft-decoding performance of these codes in the presence of additive Gaussian noise. These bounds are useful for finding large binary block codes with good performance and for verifying the performance obtained by specific soft-decoding algorithms presently under development.

I. Introduction

Reed-Solomon (RS) codes are currently used in the DSN as outer codes in a concatenated coding system. For this application, they are decoded by algebraic techniques using operations in the field over which the code is designed. An (n, k) RS code C over $GF(2^m)$ has codewords of length $n = 2^m - 1$ symbols, where each symbol is a binary m-tuple. Let A_i be the number of codewords of weight i in C, then the vector $(A_0, A_1, \cdots, A_n)$ is called the weight distribution of C, where the weight (Hamming weight) of a codeword is the number of its nonzero coordinates. The term "coordinate" assumes different meanings depending on how one views the code: One may assume that there are n coordinates, each having a value in $GF(2^m)$, or one may consider the binary expansion of the code, i.e., a binary (nm, km) code, where each coordinate is a single bit. Hence, one may be interested in the symbol weight distribution or in the binary weight distribution of a (nonbinary) code. The latter depends on the specific symbol to binary m-tuple mapping that was chosen. Which of these distributions is of interest depends on which type of decoding algorithm one plans to use, since weight distributions are essential in evaluating the error-correcting performance of a code. The symbol weight distribution of RS codes is well known [1] and can be used to find the performance of algebraic decoders working on symbols. The full error-correcting power of a code is obtained when soft, maximum-likelihood decoding is used, working directly on unquantized vectors in the nm-dimensional Euclidean space. Soft, maximum-likelihood decoding is superior to its hard quantized version by more than 2 dB. Furthermore, the algebraic decoding techniques usually employed for RS codes are not maximum-likelihood, but rather "incomplete" decoding techniques with a nonzero probability of decoding failure.

II. Binary Weight Distribution

This article focuses on evaluating the soft, maximum-likelihood decoding performance of RS codes, and therefore one needs to compute the binary weight enumerators of these codes. Such a task is a long-standing open problem in coding theory due to its intrinsic complexity. However, approximate results have been found and results for special classes of codes are known.

In general, one could think of using an exhaustive enumeration to find the numbers A_i by considering each codeword. Unfortunately, such a method is limited to fairly short codes, even on the most powerful computers available.

It was possible, for example, to find by exhaustive enumeration the weight distribution of a (21,15) binary code obtained from the (7,5) RS code over GF(2^3), but it was impractical to find that of a (60,36) binary code obtained from the (15,9) RS code over GF(2^4), since it involves 2^{36} codewords. Fortunately, a well-known result from coding theory, the MacWilliams identities [2], can be used to relate the weight distribution of a code to that of its dual. For example, one can find the binary weight distribution of the (15,9) RS code from that of its (15,6) dual code, by exhaustive enumeration on 2^{24} codewords instead of 2^{36} codewords.

Let the weight enumerator of a code C be defined as $W_C(x,y) = \sum_{i=0}^{n} A_i x^{n-i} y^i$. Then the weight enumerator of the dual code $C^\perp$ of a binary code C is given by [MacWilliams identity over GF(2)]

$$W_{C^\perp} = \frac{1}{2^k} W_C(x+y, x-y)$$

The generator polynomial of an (n,k) RS code C may be written as

$$g(x) = \prod_{i=1}^{n-k} (x - \alpha^{i+b})$$

where b can be chosen among the values $0, 1, \cdots, n-1$, and α is a root of the primitive polynomial over GF(2) defining the field GF(2^m). The parity check polynomial $h(x)$ of the code C

$$h(x) = \frac{x^n - 1}{g(x)} = \prod_{i=n-k+1}^{n} (x - \alpha^{i+b})$$

is the generator of the dual code $C^\perp$.

The binary weight distribution of the (21,15) binary code derived from the (7,5) RS code is shown in Table 1 together with the distribution of the (21,6) dual code associated with the (7,2) RS code. Results are shown for different values of the parameter b that correspond to different assignments of symbols to binary m-tuples. These are only a small subset of all possible assignments. The weight distributions shown in Table 1 could be found by exhaustive enumeration. For the (7,2) RS code, the largest binary minimum distance found was 8, which is the best possible according to [4]. For the (7,5) RS code the best result was $d_{min} = 4$, which meets the Griesmer upper bound [3].

The weight distribution of the (60,36) binary code was found by using the MacWilliams identity for binary codes, by a procedure shown in Fig. 1. First, the (15,6) dual code was generated by using the parity check polynomial of the (15,9) code as its generator. Then, the (15,6) code over GF(2^4) was represented as a binary (60,24) code by mapping symbols in GF(2^4) to binary 4-tuples by using the representation of field elements given by the irreducible polynomial $1+x+x^4$ over GF(2). The weight distribution of the (60,24) code was found by exhaustive enumeration, and finally, the weight distribution of the (60,36) code was computed by the MacWilliams identity for binary codes.

The missing arrow in the block diagram of Fig. 1 stresses the fact that the resulting (60,36) code is not necessarily related to its nonbinary parent, the (15,9) code, by the same mapping relating the (15,6) code to the (60,24) code. Table 2 shows the binary weight distributions for some (60,24) codes derived from the (15,6) RS code, where the largest minimum distance found was 13. It is known [4] that at least one (60,24) code exists for some value of d_{min} in the range 16 to 18. Table 3 shows similar results for the (60,36) code, where the largest minimum distance found was 8. At least one (60,36) code exists for some value of d_{min} in the range 9 to 12 [4].

III. Performance Evaluation

The soft decoding performance of block codes can be estimated by union bounding techniques. Specifically the word error probability P_w is upper bounded by [5]

$$P_w \leq \frac{1}{2} \sum_{j=2}^{M} \operatorname{erfc}\left(\sqrt{w_j R \frac{E_b}{N_o}}\right)$$

where $R = k/n$ is the code rate, $M = 2^k$ is the number of codewords, and w_j is the weight of the jth codeword. The bound on P_w may be easily rewritten in terms of the weight distribution A_i as

$$P_w \leq \frac{1}{2} \sum_{i=1}^{n} A_i \operatorname{erfc}\left(\sqrt{iR\frac{E_b}{N_o}}\right)$$

Similarly, for hard quantized, maximum-likelihood decoding one can derive the union bound [5]

$$P_w \leq \sum_{j=2}^{M} \left[\sqrt{4p(1-p)}\right]^{w_j}$$

where $p = \frac{1}{2}\operatorname{erfc}\left(\sqrt{R\frac{E_b}{N_o}}\right)$.

The word error probability P_w can be related to the average bit error probability P_b by observing that when at least $t+1$ errors occur, the decoder produces an errroneous codeword containing at least $d_{min} = 2t+1$ errors over n symbols. Therefore, kd_{min}/n is the average number of erroneous bits. Since in a codeword there are k bits, one has

$$P_b \approx \frac{d_{min}}{n} P_w$$

These bounds and approximations were used in Fig. 2 to evaluate the performance of the (60,36) binary code derived from the (15,9) RS code with $b = 0$.

At a high signal-to-noise ratio (SNR), the approximation $\operatorname{erfc}(x) \approx e^{-x^2}/x\sqrt{\pi}$ may be used. Considering only the contribution of codewords at d_{min}, for soft decoding, one has the approximation

$$P_w \approx \frac{1}{2} A_{d_{min}} \frac{e^{-u^2}}{u\sqrt{\pi}}$$

where $u = \sqrt{Rd_{min}E_b/N_0}$. The probability of bit error P_b may be approximated by $P_b \approx (d_{min}/n)\, P_w$, as shown in Fig. 2.

Experience with simulation results for smaller codes indicates that this approximation is usually close to the true performance, while the bounds become loose at P_b larger than 10^{-6}.

IV. Conclusion

By computing the binary weight distribution of block codes, it is possible to estimate their performance with soft, maximum-likelihood decoding. This is useful in order to find large binary block codes with good performance, and to verify the performance obtained by specific soft-decoding algorithms presently under development.

References

[1] R. Blahut, *Theory and Practice of Error Control Codes*, Reading, Massachusetts: Addison-Wesley, 1983.

[2] R. J. McEliece, *The Theory of Information and Coding*, Reading, Massachusetts: Addison-Wesley, 1977.

[3] F. J. MacWilliams and N. J. Sloane, *The Theory of Error-Correcting Codes*, New York: North-Holland Publishing Co., 1977.

[4] T. Verhoeff, "An Updated Table of Minimum-Distance Bounds for Binary Linear Codes," *IEEE Transactions on Information Theory*, vol. IT-33, no. 5, pp. 65–80, September 1987.

[5] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, New York: McGraw-Hill, 1979.

Table 1. Binary weight distributions for the (7,2) and (7,5) codes.

	(21,6) CODE			(21,15) CODE		
weight	b=0, b=1	b=2, b=6	b=3, b=4, b=5	b=0, b=4	b=1, b=2, b=3	b=5, b=6
0	1	1	1	1	1	1
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	28	21	0
4	0	0	0	84	91	210
5	0	0	0	273	322	0
6	0	0	0	924	875	1638
7	3	0	0	1956	1809	0
8	0	21	14	2982	3129	6468
9	7	0	0	4340	4585	0
10	21	0	21	5796	5551	10878
11	21	0	0	5796	5551	0
12	7	42	21	4340	4585	9310
13	0	0	0	2982	3129	0
14	3	0	7	1956	1809	3570
15	0	0	0	924	875	0
16	0	0	0	273	322	651
17	0	0	0	84	91	0
18	0	0	0	28	21	42
19	0	0	0	0	0	0
20	0	0	0	0	0	0
21	1	0	0	1	1	0

Table 2. Weight distributions of the (60,24) code.

weight	b=0, b=5	b=1, b=4	b=2, b=3	b=6, b=14	b=7, b=13	b=8, b=12	b=9, b=11	b = 10
0	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	12
11	0	0	0	0	0	0	0	0
12	0	0	15	30	30	0	15	0
13	15	75	90	0	0	0	0	0
14	150	300	180	450	375	420	390	465
15	676	859	679	0	0	0	0	0
16	2250	2160	2490	5190	4125	4530	4500	4425
17	6555	5520	5505	0	0	0	0	0
18	14720	13220	13265	23420	28760	27485	27225	27240
19	29565	29760	29955	0	0	0	0	0
20	56304	60690	60795	135420	120585	121875	123000	120204
21	113255	115460	117455	0	0	0	0	0
22	218760	206520	205410	361140	408810	407565	407565	416895
23	342285	342180	339525	0	0	0	0	0
24	493400	531470	525185	1185680	1058015	1060295	1056500	1043975
25	758583	756000	753105	0	0	0	0	0
26	1079040	1000860	1018335	1778220	2016660	2016945	2020005	2034210
27	1277425	1275280	1281835	0	0	0	0	0
28	1414125	1519215	1509690	3387720	3046005	3040095	3043830	3017910
29	1665945	1669170	1666155	0	0	0	0	0
30	1831108	1719736	1717876	3013272	3414132	3418617	3413237	3450383
31	1665945	1669170	1666155	0	0	0	0	0
32	1414125	1519215	1509690	3403485	3040170	3041160	3041205	3012720
33	1277425	1275280	1281835	0	0	0	0	0
34	1079040	1000860	1018335	1779060	2015760	2015895	2018235	2027940
35	758583	756000	753105	0	0	0	0	0
36	493400	531470	525185	1176580	1061395	1059385	1058160	1057440
37	342285	342180	339525	0	0	0	0	0
38	218760	206520	205410	360300	409950	408615	409575	411105
39	113255	115460	117455	0	0	0	0	0
40	56304	60690	60795	138168	119493	122493	122148	119361
41	29565	29760	29955	0	0	0	0	0
42	14720	13220	13265	23780	28100	27035	26375	28070
43	6555	5520	5505	0	0	0	0	0

Table 2 (contd).

weight	b=0, b=5	b=1, b=4	b=2, b=3	b=6, b=14	b=7, b=13	b=8, b=12	b=9, b=11	b = 10
44	2250	2160	2490	4890	4305	4245	4755	4350
45	676	859	679	0	0	0	0	0
46	150	300	180	390	525	495	465	480
47	15	75	90	0	0	0	0	0
48	0	0	15	20	20	65	30	30
49	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0
52	0	0	0	0	0	0	0	0
53	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0
60	1	1	1	0	0	0	0	0

Table 3. Weight distributions of the (60,36) code.

weight	b=0, b=5	b=1, b=4	b=2, b=3	b=6, b=14	b=7, b=8, b=12	b=9, b=11	b = 10	b = 13
0	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	60	0	0	15	0
8	105	360	270	105	105	75	135	60
9	0	0	0	660	765	945	1065	1005
10	9135	8067	9012	4350	4470	4500	4380	4605
11	0	0	0	20940	21045	20655	19995	20505
12	171290	170045	166730	84250	84370	84360	85950	84955
13	0	0	0	307620	308790	306720	310305	306690
14	2051130	2063850	2069655	1036980	1029780	1033080	1025820	1025910
15	0	0	0	3169396	3166006	3172656	3163509	3171106
16	17857290	17841435	17827110	8879100	8926260	8909250	8920440	8933025
17	0	0	0	23084220	23077425	23080395	23067975	23087925
18	110247955	110242255	110291800	55357350	55138110	55169540	55153100	55148985
19	0	0	0	121876260	121900185	121870485	121962285	121868505
20	499868640	499744149	499677249	248880309	249779349	249773439	249831315	249692244

Table 3 (contd).

weight	b=0, b=5	b=1, b=4	b=2, b=3	b=6, b=14	b=7, b=8, b=12	b=9, b=11	b = 10	b = 13
21	0	0	0	475905260	475911560	475934000	475793915	475896440
22	1686545400	1687429560	1687309875	846944880	843913200	843841440	843707280	844133640
23	0	0	0	1393888920	1393820040	1393856400	1393933350	1393917240
24	4299960090	4297337520	4297910160	2140496050	2148328210	2148448550	2148756230	2148052240
25	0	0	0	3094399368	3094425258	3094374858	3094295130	3094397658
26	8326857870	8331803670	8330907000	4181824860	4166545740	4166496360	4165579800	4166607690
27	0	0	0	5245474360	5245577050	5245564790	5245776110	5245426450
28	12370476540	12363639450	12364329360	6158040345	6180719145	6180621765	6182602665	6181074915
29	0	0	0	6821742120	6821661060	6821687280	6821545530	6821775660
30	14091448412	14098870268	14098595918	7076641208	7050800888	7050973648	7048404136	7050221828
31	0	0	0	6821742120	6821661060	6821687280	6821545530	6821775660
32	12370288365	12363796815	12364040385	6158040345	6180719145	6180621765	6182602665	6181074915
33	0	0	0	5245474360	5245577050	5245564790	5245776110	5245426450
34	8327053230	8331595110	8331101280	4181824860	4166545740	4166496360	4165579800	4166607690
35	0	0	0	3094399368	3094425258	3094374858	3094295130	3094397658
36	4299922280	4297446770	4297957910	2140496050	2148328210	2148448550	2148756230	2148052240
37	0	0	0	1393888920	1393820040	1393856400	1393933350	1393917240
38	1686443640	1687462200	1687212030	846944880	843913200	843841440	843707280	844133640
39	0	0	0	475905260	475911560	475934000	475793915	475896440
40	499970973	499664856	499699626	248880309	249779349	249773439	249831315	249692244
41	0	0	0	121876260	121900185	121870485	121962285	121868505
42	110224195	110285095	110300020	55357350	55138110	55169540	55153100	55148985
43	0	0	0	23084220	23077425	23080395	23067975	23087925
44	17833530	17831625	17829870	8879100	8926260	8909250	8920440	8933025
45	0	0	0	3169396	3166006	3172656	3163509	3171106
46	2071290	2066730	2063835	1036980	1029780	1033080	1025820	1025910
47	0	0	0	307620	308790	306720	310305	306690
48	166120	167845	168400	84250	84370	84360	85950	84955
49	0	0	0	20940	21045	20655	19995	20505
50	8895	8715	9000	4350	4470	4500	4380	4605
51	0	0	0	660	765	945	1065	1005
52	360	345	225	105	105	75	135	60
53	0	0	0	60	0	0	15	0
54	0	0	15	0	0	0	0	0
55	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0
60	0	0	0	1	1	1	1	1

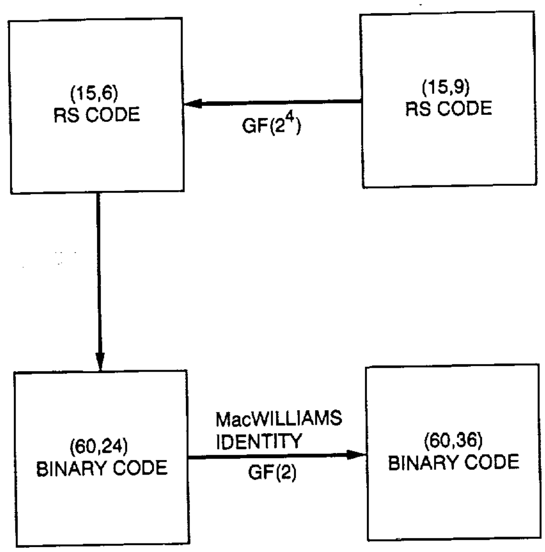

Fig. 1. Method used to find the binary weight distribution.

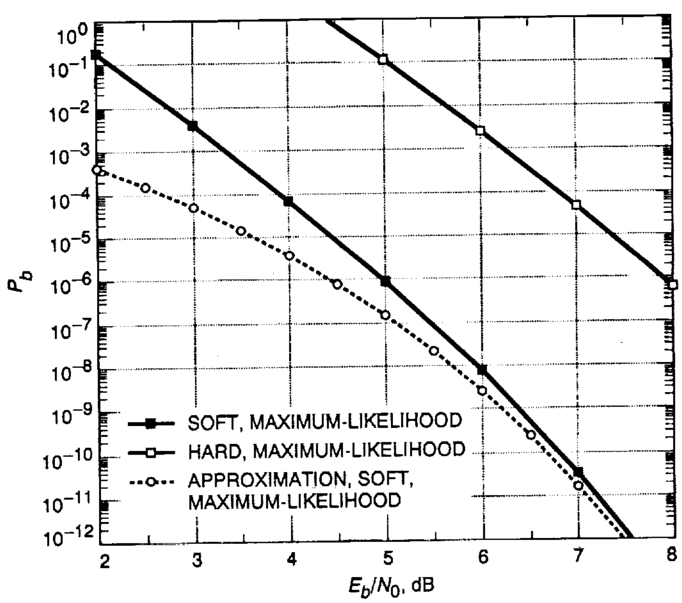

Fig. 2. Performance of (60,36) binary code derived from (15,9) RS code.