Synchronization of Reed-Solomon Codes

R. L. Miller
Communications Systems Research Section

B. B. Newman
Department of Mathematics
California Institute of Technology

Reed-Solomon codes have recently been suggested for use as an "outer" code on NASA projects, since these codes perform very well on channels prone to burst errors. This article discusses another feature of Reed-Solomon codes, viz, the way in which they can be used to acquire sync.

I. Introduction

A concatenated coding scheme, consisting of an inner (7, 1/2) convolutional code and an outer $J = 8$, $E = 16$ Reed-Solomon code, will be used for the Galileo mission and the International Solar Polar Mission, and is part of the multimission packet telemetry guidelines currently being proposed. This report examines the synchronization capabilities of Reed-Solomon codes when an appropriate coset of the code is used instead of the code itself. In this case an E-error correcting Reed-Solomon code is transformed into a new code capable of determining that there are m symbols out of sync, if e symbol errors occurred, whenever $m + e < E$. In the event that $m = 0$, i.e., the word is in sync, then the decoder will correct any pattern of $E - 1$ or fewer symbol errors.

The key idea to achieving synchronization is to use a coset of the code instead of the code itself. (A coset is obtained by adding the same vector to every code word.) From an error-correcting point of view, the coset is equivalent to the code itself. In addition, synchronization can sometimes be achieved as well. The algorithm to be presented differs from usual coding algorithms in that the information is encoded into one code, but decoded in a larger (different) code. The larger code contains the coset of the smaller code.

II. Synchronization Algorithm

Suppose that C_1 and C_2 are Reed-Solomon codes of length n whose symbols lie in $GF(q)$. Let

$$C_1 = (g_1(x)), \text{ where } g_1(x) = \prod_{i=2}^{2E-1} (x - \alpha^i), \text{ and } C_2 = (g_2(x)),$$

where

$$g_2(x) = \prod_{i=1}^{2E-1} (x - \alpha^i),$$

and α is a primitive nth root of unity. (Observe that $C_2 \subseteq C_1$.) Then the following algorithm will insure that any transmitted code word of C_2 will be resynchronized if it is received out of sync by m symbols, and e errors were made, provided that $m + e < E$. If $m = 0$, then any combination of $E - 1$ or fewer symbol errors will be corrected.

Algorithm:

Encoder

(1) Encode the information into $w(x) \in C_2$.

(2) Transmit $w(x) + g_1(x)$ (a coset).

Decoder

(1) Receive $r(x)$, which has e errors and is out of sync by m symbols.

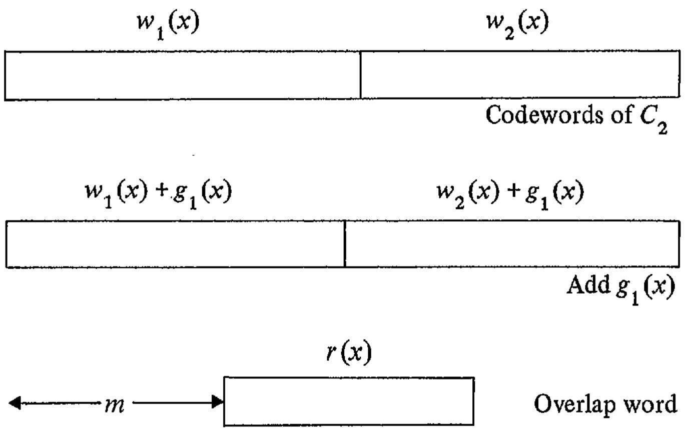
(2) Compute $r(x) - g_1(x)$.

(3) Determine m.

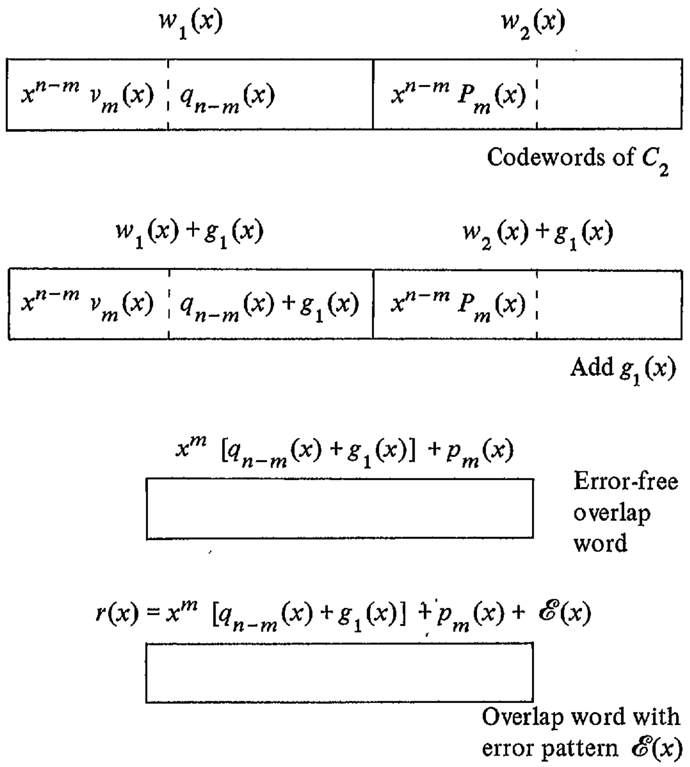
(4) If $m = 0$, then correct $r(x)$ using the C_1 – decoder. Otherwise, shift $r(x)$ ahead by $n - m$ symbols to acquire sync.

III. Verification of Algorithm Correctness

Suppose that $w_1(x)$, $w_2(x) \in C_2$ are codewords and that $w_1(x) + g_1(x)$, $w_2(x) + g_1(x)$ are transmitted over a noisy channel. Assume also that the decoder does not know where a codeword begins, and that it begins to decode m symbols out of sync. This can be represented by the following diagram.



Let $x^{n-m}\ P_m(x)$ be the prefix of $w_2(x)$ appearing in $r(x)$, where deg $P_m(x) < M$. Also let $q_{n-m}(x)$ be the suffix of $w_1(x)$ appearing in $r(x)$, where deg $q_{n-m}(x) < n$. Finally, denote $x^{n-m}\ v_m(x)$ as the prefix of $w_1(x)$ preceding $r(x)$. Then the following diagram is useful.



At stage 2, the decoder attempts to strip $g_1(x)$ from the received word by subtraction. This will succeed if the received word is in sync, i.e., if $m = 0$. Otherwise the received word is further perturbed from the originally transmitted word. This is indicated below.

$$r(x) - g_1(x) = (x^m - 1)g_1(x) + x^m q_{n-m}(x) + P_m(x) + \mathscr{E}(x)$$

Since $w_1(x), g_1(x) \in C_1$, $(x^m - 1)g_1(x) + x^m w_1(x) \in C_1$.

Thus $r(x)$ and $r(x) - [(x^m - 1)g_1(x) + x^m w_1(x)]$ have the same error pattern, $\mathscr{E}(x)$, as far as the C_1 – decoder is concerned. But,

$$\begin{aligned} r(x) - [(x^m - 1)g_1(x) + x^m w_1(x)] \\ &= x^m q_{n-m}(x) + P_m(x) - x^m w_1(x) + \mathscr{E}(x) \\ &= x^m [q_{n-m}(x) - w_1(x)] + P_m(x) + \mathscr{E}(x) \\ &= x^m [-x^{n-m}\ v_m(x)] + P_m(x) + \mathscr{E}(x) \\ &= [P_m(x) - v_m(x)] + \mathscr{E}(x), \text{ since } x^n = 1. \end{aligned}$$

Now $P_m(x)$ and $v_m(x)$ both have degree $< m$; hence $P_m(x) - v_m(x)$ has at most m nonzero terms. If $\mathscr{E}(x)$ has at most $E - m$ nonzero terms, i.e., if at most $E - m$ transmission errors occurred, then the C_1 — decoder will determine $\hat{\mathscr{E}}(x) = P_m(x) - v_m(x) + \mathscr{E}(x)$ as the error polynomial. Of course, what is desired is to compute $\mathscr{E}(x)$ and m separately. Note that if $m = 0$, then $\hat{\mathscr{E}}(x) = \mathscr{E}(x)$, and $r(x) - \mathscr{E}(x)$ is the desired codeword. Otherwise,

$$\begin{aligned} r(x) - \hat{\mathscr{E}}(x) &= x^m [q_{n-m}(x) + g_1(x)] + v_m(x) \\ &= x^m q_{n-m}(x) + v_m(x) + x^m g_1(x) \\ &= x^m [q_{n-m}(x) + x^{n-m} v_m(x)] + x^m g_1(x) \\ &= x^m [w_1(x) + g_1(x)]. \end{aligned}$$

Thus

$$\begin{aligned} r(\alpha) - \hat{\mathscr{E}}(\alpha) &= \alpha^m [w_1(\alpha) + g_1(\alpha)] \\ &= \alpha^m g_1(\alpha), \text{since } w_1 \in C_2. \end{aligned}$$

Since $g_1(x)$ is a fixed polynomial, m can be easily determined by a table look-up. Once m is known, then a shift of n-m symbols will reacquire sunc.

IV. Example

Let C_1 be the Reed-Solomon code, RS(255,225), generated by

$$g_1(x) = \prod_{i=2}^{31} (x - \alpha^i),$$

where α is a primitive 255th root of unity in GF(2^8). Let C_2 be the Reed-Solomon code, RS(255,224) generated by

$$g_2(x) = \prod_{i=1}^{31} (x - \alpha^i).$$

The information is encoded in C_2, and $g_1(x)$ is added to the parity symbols. If $r(x)$ is received out of sync by m symbols with symbol errors, then m will be determined if $m + e < 16$. Moreover, if $m = 0$, then $r(x)$ will be correctly decoded, provided that 15 or fewer symbol errors occurred.

V. System Considerations

The (255,223) 16-error correcting Reed-Solomon code has been given as a quasi-standard by various groups in NASA. If one wants to be able to acquire sync using the Reed-Solomon code alone, then additional system considerations are needed.

First, the information is encoded in one code, but decoded in another. Second, to maintain the same information rate, only 15 errors can be corrected; on the other hand, keeping the same error-correcting capability forces the information rate to drop. This memo alludes to using the former option, since initial performance studies indicate that the (255,225) 15-error correcting code performs better than the (255,223) 16-error correcting code. Of course, what have to be estimated are the expected time to acquire sync, the probability of false sync, and the buffer requirements, assuming that the ratio of the decoder's speed to the data rate is some constant. This work as well as estimating the performance comparison (dB) of using this new Reed-Solomon scheme to acquire sync vs using the (255,223) Reed-Solomon code and a fixed sync pattern should be performed in the future.

Acknowledgment

The authors wish to thank Dr. E. C. Posner for his suggestion to perform this investigation.