Design of a Quick-Look Decoder for the DSN (7, 1/2) Convolutional Code

C. A. Greenhall and R. L. Miller
Communications Systems Research Section

In a previous article, the authors showed that the DSN (7,1/2) convolutional code could be decoded by a simple "quick-look" method that requires only two shift registers of length 7 for the incoming hard-quantized channel symbols. Such a decoder is being developed for Project Galileo to circumvent an incompatibility between the DSN and the Tracking and Data Relay Satellite System (TDRSS), whose versions of the (7,1/2) code differ in the order of the symbols in each pair. Because signal-to-noise ratios are high during the near-Earth phase of the Galileo mission, quick-look decoding is feasible. The present article derives design parameters and performance figures for the three functions of the decoder: acquisition of node synchronization, generation of the decoded bits, and estimation of channel quality.

I. Introduction

The quick-look decoder discussed here is proposed for a 1200 bit/s telemetry link that will operate while the TDRSS is relaying the signal from the Galileo spacecraft. Figure 1 shows the environment of the proposed decoder. At present, the maximum likelihood (Viterbi) decoder of the TDRSS cannot be used because it expects the symbols in each pair to be reversed relative to the DSN convention. Our proposal is to transmit the hard-quantized symbols (most significant bits) from the 3-bit quantized symbols via the Ground Communications Facility (GCF) to the Mission Control and Computing Center (MCCC) at JPL, where the quick-look decoder would be implemented in software.

We explained the concept of quick-look decoding in Ref. 1. Here, we give a concrete design of such a decoder for the (7,1/2) code. Besides generating the decoded bits, the decoder also acquires node synchronization and estimates channel quality. Working from stochastic models for the channel symbol errors and information bits, we choose the parameters and compute the performance of the algorithms for these services. The parity check bits discussed briefly in Ref. 1 play a crucial role, analogous to the role of the branch metrics in Viterbi decoding. A simple error correction method, not anticipated in Ref. 1, lowers the bit error rate from $7p_e$ to $133p_e^2$, where p_e is the symbol error rate.

The following display recapitulates the encoding and decoding processes given in Ref. 1. All arithmetic is performed modulo 2.

Quantity	Sequence or vector	Formal power series
1. Information bit	b_n	$B(x) = \Sigma b_n x^n$
2. Code vectors	$C_1 = 1011011$	$C_1(x) = 1 + x^2 + x^3 + x^5 + x^6$
	$C_2 = 1111001$	$C_2(x) = 1 + x + x^2 + x^3 + x^6$
3. Channel symbols	s_{1n}, s_{2n}	$S_i(x) = \Sigma s_{in} x^n$ $= C_i(x) B(x), i = 1, 2$
4. Received symbols	s^*_{1n}, s^*_{2n}	$S^*_i(x) = \Sigma s^*_{in} x^n, i = 1, 2$
5. Inversion vectors	$A_1 = 00101$	$A_1(x) = x^2 + x^4$
	$A_2 = 11111$	$A_2(x) = 1 + x + x^2 + x^3 + x^4$
6. Decoded bit	b^*_n	$B^*(x) = \Sigma b^*_n x^n$ $= A_1(x)S^*_1(x) + A_2(x)S^*_2(x)$
7. Parity check bit	p_n	$P(x) = \Sigma p_n x^n$ $= C_2(x)S^*_1(x) + C_1(x)S^*_2(x)$

Reference 1 does not discuss symbol or bit errors. Here, we let

$$E_i(x) = \Sigma e_{in} x^n = S^*_i(x), - S_i(x), (i = 1, 2)$$

$$D(x) = \Sigma d_n x^n = B^*(x) - B(x),$$

be the formal power series for the symbol errors

$$e_{1n} = s^*_{1n} - s_{1n}, \quad e_{2n} = s^*_{2n} - s_{2n},$$

and the decoded bit errors

$$d_n = b^*_n - b_n .$$

The identities

$$C_2(x)C_1(x) + C_1(x)C_2(x) = 0,$$

$$A_1(x)C_1(x) + A_2(x)C_2(x) = 1,$$

give the parity bits and decoded bit errors in terms of the symbol errors:

$$P(x) = C_2(x)E_1(x) + C_1(x)E_2(x), \qquad (1)$$

$$D(x) = A_1(x)\,E_1(x) + A_2(x)\,E_2(x). \qquad (2)$$

These formulas are basic for the analysis of the decoder.

In the DSN version of this code, the first symbols s_{1n} are inverted by the encoder, i.e., s_{1n} (DSN) = s_{1n} + 1 (mod 2), to provide enough symbol transitions for the symbol synchronizer. The formula for computing the decoded bits (line 6 of the display) is unchanged because A_1 has an even number (2) of ones. The parity bits obtained from line 7 have to be inverted because C_2 has an odd number (5) of ones. Each parity bit p_n is the sum of 5 of the s_{1k} and 5 of the s_{2k}. If each s_{1k} is inverted then p_n is inverted.

II. Decoding With Correction of Isolated Errors

The basic decoder described in Ref. 1 does no error correction. If symbol errors are rare, however, then most of them can easily be recognized from the parity stream. If $E_1(x) = 1$, $E_2(x) = 0$ (i.e., s^*_{10} is wrong and all other symbols are right), then from Eqs. (1) and (2) we have $P(x) = C_2(x)$, $D(x) = A_1(x)$. The pattern $C_2 = 1111001$ appears in the parity stream p_n, and 00101 in the bit error stream d_n. Thus, if we see the C_2 pattern from time $n - 6$ to time n in the parity stream, then, presuming that $e_{1,n-6} = 1$, we correct the last few decoded bits by adding the vector A_1 to them, modulo 2. Similarly, if we see C_1 in the parity stream we correct the decoded bits by adding A_2. We formalize this as follows:

Correction Algorithm. Keep the last 7 parity bits and decoded information bits in shift registers

$$P = (p_{n-6}, p_{n-5}, \ldots, p_n),$$

$$B^* = (b^*_{n-6}, b^*_{n-5}, \ldots, b^*_n).$$

For each bit time n

If P = 1011011 then let $B^* = B^*$ + 1111100

If P = 1111001 then let $B^* = B^*$ + 0010100

Pass b^*_{n-6} to the calling program.

This algorithm corrects all bit errors due to isolated symbol errors; a symbol error at time n is said to be *isolated* if there are no other symbol errors from time $n - 6$ to $n + 6$, inclusive. Most symbol errors are isolated. Bit errors are caused by "bursts" of two or more symbol errors. The derivation below yields the following performance estimates: Assume that the

symbol errors are independent, with probability p_e of occurring. (This is the average number of symbol errors per *symbol*, not per bit.) Then the bit error probability p_d satisfies

$$133p_e^2\,(1 - 36p_e) < p_d < 133p_e^2 \qquad (3)$$

for $p_e \leqslant 10^{-2}$. We shall use the upper bound for computation.

Figure 2 plots the bit error rate (BER), p_d, of the quick-look decoder, and the symbol error rate (SER), p_e, as functions of the bit signal-to-noise ratio E_b/N_o. Formulas used are Eq. (3), together with

$$p_e = Q\left(\sqrt{E_b/N_o}\right),$$

where $Q(x)$ is the probability that a standard Gaussian is greater than x.

According to our present information about the Galileo-TDRSS link, the E_b/N_o, including receiver and demodulation losses, will be 11.3 dB. From Fig. 2 we see that the quick-look decoder margin is 0.5 dB for a BER of 10^{-5}, and 3.3 dB for a BER of 5×10^{-3}.

To derive the estimate (3), we introduce the following assumptions and notation: At time zero, the decoder shift registers are free of symbol errors. The symbol errors e_{1n}, e_{2n}, $n \geqslant 1$, are independent with probability p_e of being 1 and $1 - p_e$ of being 0. The notation e_n designates the pair (e_{1n}, e_{2n}), and $e_n = 0$ means $e_{1n} = 0, e_{2n} = 0$.

Definition. A (symbol error) *burst* is said to occur between bit times m and n $(m \leqslant n)$ if

(1) $e_m \neq 0, e_n \neq 0$;

(2) $e_k = 0, m - 6 \leqslant k \leqslant m - 1, n + 1 \leqslant k \leqslant n + 6$;

(3) $(e_m \ldots, e_n)$ contains no run of 6 consecutive e_k equal to 0.

For $1 \leqslant m \leqslant 6$ a burst can start at m provided all previous bit times are free of symbol errors.

A burst having r symbol errors is called an *r-burst*. A 1-burst is also called an *isolated symbol error.*

Every symbol error in the sequence e_n belongs to some burst. The correction algorithm corrects all bit errors due to isolated symbol errors. We shall use Feller's theory of recurrent events (Ref. 3) to study the bit error rate due to longer bursts.

Definition. The event ξ is said to occur at bit time n, $n \geqslant 7$, if time $n - 6$ is the end of a burst. In other words, $e_{n-6} \neq 0$ and $e_k = 0$ for $n - 5 \leqslant k \leqslant n$.

Each burst is associated with an occurrence of ξ. When ξ occurs, the decoder shift registers are free from symbol errors except in the last position. In the next bit time, e_{n-6} is shifted out of the registers and the situation is exactly like that at time 1. Each time ξ occurs, the process "starts from scratch."

One can prove that ξ is a persistent, aperiodic recurrent event in the sense of Feller. The times between occurrences of ξ are independent random variables with a common distribution. Let M denote the *mean recurrence time*, and let u_n denote the probability that ξ occurs at time n. Then $u_n \to 1/M$ as $n > \infty$. Hence, we can interpret lim u_n as the *average number of bursts per bit.*

Furthermore, let β denote an arbitrary class of burst types (a burst *type* is the finite sequence of zeros and ones that make up a burst, considered without regard to when the burst occurs), e.g., the 2-bursts, or the bursts of length 10 bit times. Define $\xi(\beta)$ like ξ except that the type of the burst that ends at time $n - 6$ must belong to β. Then $\xi(\beta)$ is also a recurrent event. Let $u_n(\beta)$ be the probability that $\xi(\beta)$ occurs at time n. If ℓ is an upper bound for the lengths of the burst types in β, then $u_n(\beta) = u(\beta)$, a constant, for all $n \geqslant \ell + 12$.

The bit error rate p_d of the decoder can now be written

$$p_d = \sum_{r=2}^{\infty} p_d(r),$$

where $p_d(r)$ is the bit error rate due to r-bursts. (Recall that $p_d(1) = 0$.) As expected, for small p_e the first term $p_d(2)$ dominates the rest. There are 25 individual types of 2-bursts, call them $\beta_1, \ldots, \beta_{25}$ (in no particular order). They are diagrammed below:

1st symbol:	1 01	0000001
2nd symbol:	1 '10	'1000000 '
	10	1000000
	01	'0000001 '
	11	1000001
	00	'0000000 '
	00	0000000
	11	'1000001 '

The bit error rate due to 2-bursts is given by

$$p_d(2) = \sum_{k=1}^{25} x_k u(\beta_k),$$

where x_k is the number of bit errors allowed by the decoder when it encounters a burst of type β_k.

Because the 2-burst types are at most 7 bits long, we have $u_n(\beta) = u(\beta)$ for $1 \leqslant k \leqslant 25$ and $n \geqslant 19$. Because each burst is surrounded by 12 good symbols before and after, and contains up to 12 good symbols in its interior, we have

$$u(\beta_k) = (1 - p_s)^{m_k} p_s^2,$$

where $24 \leqslant m_k \leqslant 36$.

Two examples of the computation of x_k are shown below:

Burst type	Parity errors	Bit errors	x_k
1001	1111001	00101	
0000	1111001	00101	
	1110110001	00101101	4
10000	1111001	00101	
00001	1011011	11111	
	1 1111001 011	0010100	
	C_2	001100111	5

In the second example, the pattern C_2 appears "by accident" in the parity stream, starting at bit time 2. At bit time 8 the correction algorithm, behaving as if an isolated first-symbol error had occurred at time 2, adds A_1 to the decoded bit shift register. Notice that this spurious "correction" merely redistributes the bit errors without increasing their number. This is the only type of 2-burst in which C_1 or C_2 appears in the parity stream.

We find that

$$\sum_{k=1}^{25} x_k = 133,$$

and hence

$$133p_e^2 (1 - p_e)^{36} < p_d(2) < 133p_e^2 (1 - p_e)^{24} \qquad (4)$$

We now estimate the contribution of r-bursts, $r > 2$, to the bit error rate. The effect of the correction algorithm will be neglected, for we just saw that it has no effect on the 2-burst performance. An r-burst begins with the symbol pair (1,1), (1,0), or (0,1). Then there are 0 to 5 good bit times, followed by an $(r - 2)$-burst (for the (1,1) case) or an $(r - 1)$-burst (for the other two cases). Therefore, the number of r-burst types, y_r, satisfies

$$y_r = 6y_{r-1} + 12y_{r-2}, \quad r \geqslant 3$$

$$y_1 = 2, y_2 = 25.$$

The solution of this initial value problem is

$$y_r = K_1 z_1^r + K_2 z_2^r,$$

where $z_1 = 3 + \sqrt{21}, z_2 = 3 - \sqrt{21}, K_1 = (273 - 15\sqrt{21})/504$, $K_2 = (273 + 15\sqrt{21})/504$. The rate of occurrence of each type of r-burst is at most $p_e^r (1 - p_e)^{24}$, and each of the r symbol errors gives rise to at most 5 bit errors. Therefore,

$$p_d(r) < y_r \cdot 5r \cdot p_e^r (1 - p_e)^{24}. \qquad (5)$$

Summing (5) over $r \geqslant 3$ and adding the upper bound for $p_d(2)$ from (4), we obtain an expression that remains below $133\,p_e^2$ for $p_e \leqslant 10^{-2}$. The lower bound in (3) comes from the lower bound in (4) for $p_d(2)$.

In view of the crudity of (5), we conjecture that the upper bound in (3) remains valid even when the effect of the correction algorithm on r-bursts ($r > 2$) is included.

III. Node Synchronization

The decoder receives the stream of corrupted symbols

$$s_{11}^*, s_{12}^*, s_{21}^*, s_{22}^*, s_{31}^*, s_{32}^*, \ldots$$

and must decide how to pair them off. We use the method of the *up-down counter*, also used in Viterbi decoders (Ref. 2). In our case, the counter is driven by the parity bits. If a parity bit is zero, the counter is decremented by 1; if a parity bit is 1, the counter is incremented by $k - 1$, where k is a fixed integer $\geqslant 2$. The counter starts at zero and is not allowed to become negative. If it ever reaches a preassigned threshold T then it is reset to zero, the decoder lets one symbol go by, and the decoder informs the calling program that a resync has occurred.

The problem here is to choose k and T. We must choose k so that the average drift rate of the counter is negative for true sync and positive for false sync. If sync is true, then Eq. (1) holds. The counter rarely departs from zero if the SER is low, because parity errors are rare. Eventually, though, it does reach the threshold, causing a *false alarm*, a needless loss of sync. We want the expected time to false alarm, E_{fa}, to be large. To be specific, we shall set T high enough so that E_{fa} is more than 100 times as long as Galileo will be using the TDRSS. To do this, it is necessary to assume a certain cleanliness of the channel. Our *symbol error assumption* is

$$\text{SER} \leqslant 6.13 \times 10^{-3} \tag{6}$$

at which point $E_b/N_o = 8$ dB, BER $= 5 \times 10^{-3}$ (see Fig. 2).

Let sync be false. Then one expects a relatively high density of parity errors, which force the counter to rise quickly to the threshold. For this to happen, however, it is essential that the information bit stream contain enough information. If, for example, the information bits are all zero, then most of the corrupted symbols and parity bits are zero, and one cannot tell true sync from false. Our *information bit assumptions* are (1) the ratio of ones to total bits is between 5 percent and 95 percent, and (2) the ratio of transitions to total bits is at least 5 percent. Incorporating these assumptions into stochastic models, we can estimate the expected time to resync the decoder, E_{rs}, which we want to be small. The two quantities E_{fa} and E_{rs} specify the performance of the sync algorithm.

The symbol error and bit assumptions define a *design point*, from which the optimization procedure of subsection C below yields the counter parameters

$$k = 8, \quad T = 512.$$

The performance of the algorithm is

$$E_{fa} > 10^9 \text{ bits}, \quad E_{rs} = 460 \text{ bits}.$$

Our design point is negotiable; if the above choice is unsuitable, the authors are willing to recompute k and T.

A. True Sync

Let us assume that node sync is true and that the symbol errors e_{1n}, e_{2n} are independent with probability p_e of being one. Our aim is to get a lower bound on E_{fa}. We begin with a preliminary remark. As we saw in section II, most of the symbol errors are isolated, and each isolated symbol error produces 5 parity errors. Therefore, the number of parity errors n_p in a given stretch of n_b bits is approximately 5 times the number of symbol errors n_e. Moreover, Eq. (1) implies that $n_p \leqslant 5n_e$ provided that the decoder shift registers are initially clean, which, henceforth, we assume.

The up-down counter executes a random walk with a reflecting barrier at -1 and an absorbing barrier at T. Since the steps are not independent it is convenient to bound the motion of this counter, call it Counter 1, by the motion of another (fictitious) counter, Counter 2, that executes a random walk with independent steps. Counter 2 operates each *symbol* time as follows: If the next symbol is correct then the counter is decremented by 1/2; if it is incorrect then the counter is incremented by $5k - (1/2)$. There is a reflecting barrier at $-1/2$ and an absorbing barrier at T. Without barriers, in n_b bit times Counter 1 moves up by $(k-1)n_p - (n_b - n_p) = kn_p - n_b$ (probably negative), while Counter 2 moves up by $[5k - (1/2)]\, n_e - (1/2)(2n_b - n_e) = 5kn_e - n_b$. It follows from our earlier remark that Counter 1 advances no more than Counter 2, with or without barriers.

The difference equation method given in Chapter XIV of Ref. 3 can be used to derive bounds on the mean first-passage time of the absorbing barrier T by Counter 2. Using the conventions of Ref. 3, we consider a random walk on the integers with a reflecting barrier at zero and an absorbing barrier at a. It advances by d with probability p and by -1 with probability $q = 1 - p$. The parameters are given by

$$a = 2T + 1, \quad d = 10k - 1, \quad p = p_e. \tag{7}$$

Let D_j be the mean first-passage time for a walk that starts j units above the zero-level. We assume, but do not prove, that D_j is finite. Then, it satisfies the difference equation

$$D_j = pD_{j+d} + qD_{j-1} + 1, \quad 1 \leqslant j \leqslant a - 1, \tag{8}$$

with boundary conditions

$$D_0 = D_1,$$

$$D_j = 0, \quad a \leqslant j \leqslant a + d - 1$$

Our aim is to find bounds for D_1 (see (16) and (17) below). The characteristic equation of Eq. (8) is

$$pz^d + qz^{-1} = 1, \tag{9}$$

which, provided $q - pd \neq 0$, has a simple root at 1 and just one other positive root λ. For any A and B, the sequence

$$E_j = \frac{j}{q - pd} + A + B\lambda^j \tag{10}$$

satisfies

$$E_j = pE_{j+d} + qE_{j-1} + 1 .$$

Solving for the A and B that make

$$E_0 = E_1 , \quad E_a = 0 , \tag{11}$$

we get

$$E_j = \frac{1}{q - pd} \frac{\lambda^a - \lambda^j}{\lambda - 1} + j - a \tag{12}$$

The sequence $-E_j$ is convex. This, together with Eq. (11), implies $E_j < 0$ for $j > a$. Therefore, the difference $\Delta_j = D_j - E_j$ satisfies

$$\Delta_j = p\Delta_{j+d} + q\Delta_{j-1} , \quad 1 \leqslant j \leqslant a - 1 , \tag{13}$$

$$\Delta_0 = \Delta_1 , \tag{14}$$

$$\Delta_j \geqslant 0 , \quad a \leqslant j \leqslant a + d - 1 . \tag{15}$$

We claim that $\Delta_j \geqslant 0$ for $0 \leqslant j \leqslant a + d - 1$. To prove this, let Δ_j assume its minimum value m at $j = r$. Because of Eq. (14) we can assume $r \geqslant 1$. Repeated use of Eq. (13) gives $\Delta_{r+d} = m$, $\Delta_{r+2d} = m$, and finally $\Delta_s = m$ for some s such that $a \leqslant s \leqslant a + d - 1$. This proves $m \geqslant 0$.

Thus we have the result

$$D_j \geqslant E_j , \quad 0 \leqslant j \leqslant a + d - 1 .$$

In particular

$$D_1 = D_0 \geqslant \frac{1}{q - pd} \left(\frac{\lambda^a - 1}{\lambda - 1} - a \right) . \tag{16}$$

A similar argument that replaces (11) by

$$E_0 = E_1 , \quad E_{a+d-1} = 0 ,$$

leads to the upper bound

$$D_1 \leqslant \frac{1}{q - pd} \left(\frac{\lambda^{a+d-1} - 1}{\lambda - 1} - (a + d - 1) \right) . \tag{17}$$

For design, we use the lower bound (16), plus (7) and the bound

$$E_{fa} \geqslant \frac{1}{2} D_1 , \tag{18}$$

which follows from the relationship between Counters 1 and 2.

B. False Sync

If node sync is false then we want the up-down counter to rise quickly to the threshold T. We can neglect symbol errors because they produce more parity errors, which make the counter rise even faster.

The encoder transmits the symbols serially in a stream represented by the formal power series

$$S_1(x^2) + xS_2(x^2) . \tag{19}$$

The decoder sends the coefficients of $1, x^2, x^4, \ldots$ down the first-symbol pipe, and the coefficients of $x, x^3, x^5, \ldots$ down the second-symbol pipe. If sync is false, then the decoder sees the stream whose formal power series is x times (19), or

$$xS_1(x^2) + x^2S_2(x^2) . \tag{20}$$

Therefore, it sends $xS_2(x)$ down the first-symbol pipe, and $S_1(x)$ down the second-symbol pipe. The parity stream comes out as

$$\begin{aligned} P(x) &= C_2(x)xS_2(x) + C_1(x)S_1(x) \\ &= [xC_2(x)^2 + C_1(x)^2]\, B(x) \\ &= K(x)B(x) \end{aligned} \tag{21}$$

where

$$K(x) = xC_2(x^2) + C_1(x^2)$$

$$= 1 + x + x^3 + x^4 + x^5 + x^6 + x^7 + x^{10} + x^{12} + x^{13}.$$

(Note that $f(x)^2 = f(x^2)$ (mod 2) for any polynomial $f(x)$.) Each parity bit is the modulo 2 sum of 10 information bits. The density of parity errors depends on the distribution of zeros and ones in the information bit stream. We shall use two models for the b_n.

Model B. Independent bits. Let the b_n be independent with probability p_b of being one, $q_b = 1 - p_b$ of being zero (p_b is *not* a bit *error* probability). The density of parity errors, μ_p, is the probability that the sum of 10 of the b_n is odd. By considering the expansions of $(q_b + p_b)^{10}$ and $(q_b - p_b)^{10}$ we get

$$\mu_p = \frac{1}{2}[1 - (q_b - p_b)^{10}] =: \mu_{pb} \tag{22}$$

Model T. Independent transitions. Let $t_n = b_n + b_{n-1}$ (mod 2) be independent, with probability p_t of being one, $q_t = 1 - p_t$ of being zero. (The probability that b_n is one is 1/2.) Let $T(x)$ be the corresponding formal power series. Then

$$T(x) = \sum t_n x^n = (1 + x)B(x).$$

Since $K(1) = 0$ we have $K(x) = (1 + x)L(x)$ and

$$P(x) = L(x)T(x),$$

where

$$L(x) = 1 + x^3 + x^5 + x^7 + x^8 + x^9 + x^{12}$$

Because $L(x)$ has 7 nonzero terms, the density of parity errors is now

$$\frac{1}{2}[1 - (q_t - p_t)^7] =: \mu_{pt} \tag{23}$$

Let $p_b = p_t \leq 1/2$. Then the bit streams in both models carry the same amount of information, and $\mu_{pt} \leq \mu_{pb}$. Hence, in order to be conservative about the upward drift of the counter, we use model T to compute μ_p. The average drift rate of the counter is $(k - 1)\mu_p - (1 - \mu_p) = k\mu_p - 1$, which must be positive if the counter is to advance rapidly and steadily to the threshold. We shall not be far wrong, then, if we estimate the expected time to resync, E_{rs}, by

$$E_{rs} = \frac{T}{k\mu_p - 1}, \tag{24}$$

where $\mu_p = \mu_{pt}$.

C. Choice of Sync Parameters

We wish to make a rational choice of the counter parameters k and T. Recall that our chosen design point is defined by

$p_e = 6.13 \times 10^{-3}$ symbol error rate

$p_t = 0.05$ bit transition rate.

The decoder would be used by the Galileo mission for at most one hour, about 4×10^6 bits at 1200 b/s. Let us require that the expected time to false alarm, E_{fa}, be much greater, say 10^9 bits. Keeping p_e and p_t fixed we try a value of k. By trial and error we find the least T such that the lower bound in (18) for E_{fa} exceeds 10^9. Then the expected time to resync, E_{rs}, is obtained from Eq. (24). This gives E_{rs} as a function of k. If k is too large, then the counter under true sync has less negative drift; hence T must be made large in order to make $E_{fa} > 10^9$, and this makes E_{rs} large. If k is too small, then T can be made smaller for a given E_{fa}, but E_{rs} still becomes large because the denominator in Eq. (24) is close to zero. There is a k that minimizes E_{rs}; for our design point we get $k = 8$, $T = 511$, $E_{rs} = 471$. The contribution from false alarms to bit error rate is $E_{rs}/E_{fa} > 4.7 \times 10^{-7}$.

Of course, if the channel is actually better than our design point, then E_{fa} is much greater; for example, if $p_e = 10^{-3}$ then $E_{fa} > 9 \times 10^{22}$. On the other hand, if $p_e = 10^{-2}$ then $1.1 \times 10^5 < E_{fa} < 1.9 \times 10^5$ (from the theory of Counter 2), so that we should expect a false alarm every 2 minutes on the average.

IV. Estimation of Channel Quality (SER)

Assume that node sync is true. As we pointed out earlier, the number of parity errors n_p in n_b bits is about $5n_e$, where n_e is the number of symbol errors. Therefore, the symbol error rate p_e can be estimated from n_p by

$$p_e \approx \frac{n_e}{2n_b} \approx \frac{n_p}{10n_b}. \tag{25}$$

To use Eq. (25), one could wait until $n_p = 5n$ where n is a fixed integer, say 25. Then the relative one-sigma error in the estimator Eq. (25) is about $1/\sqrt{n}$. If p_e is low, say 10^{-6}, then symbol errors occur only once every 7 minutes; thus, it would take too long to accumulate 25 of them. To accommodate a wide range of SER, we propose the following scheme. Assume that the decoder shares memory with a "main program." The locations n_p, n_b, and p_e are available to both programs. Initially, all three locations are set to zero.

The *decoder* accumulates the parity counts in n_p and bit counts in n_b. When n_p reaches $5n$, the decoder sets $p_e = n_p/(10n_b)$, $n_p = 0$, $n_b = 0$, and begins counting again. We suggest $n = 25$.

The *main program*, whenever it wants an SER estimate, checks p_e. If $p_e \neq 0$ it uses p_e as the estimate and sets $p_e = 0$. If $p_e = 0$ it computes $n_p/(10n_b)$ for itself as the estimate.

In other words, the main program takes the decoder's most recent update, if one has been provided since the main program's last inquiry; otherwise, the main program does the best it can with the most recent bit counts. The decoder updates p_e once every $n/(2p_e)$ bits, on the average.

V. Conclusions

We have derived design parameters and performance of a simple quick-look decoder for the DSN (7, 1/2) convolutional code. The decoder uses parity check bits for correcting isolated errors, finding node sync, and estimating the symbol error rate. An up-down counter, driven by the parity bits, is used to detect false node sync. The performance margin for near-Earth Galileo telemetry is estimated to be 3.3 dB for a bit error rate of 5×10^{-3}.

References

1. Greenhall, C. A., and Miller, R. L., "Quick-Look Decoding Schemes for DSN Convolutional Codes," *DSN Progress Report 42-51*, pp. 162-166, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 15 June 1979.
2. Gilhousen, K. W., et al., *Coding Systems Study for High Data Rate Telemetry Links*, Linkabit Corporation, San Diego, California, January 1971.
3. Feller, W., *Introduction to Probability Theory and Its Applications*, Volume I, 3rd Edition, New York, 1968.

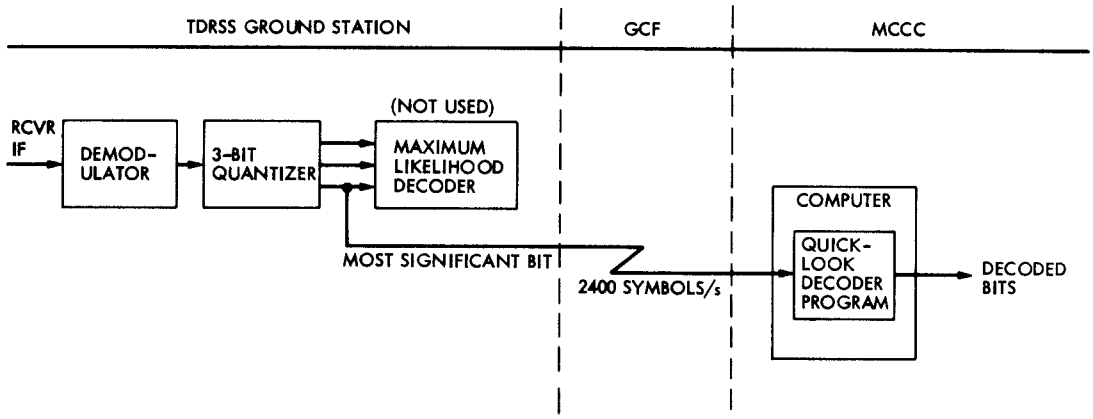

Fig.1. Environment of quick-look decoder

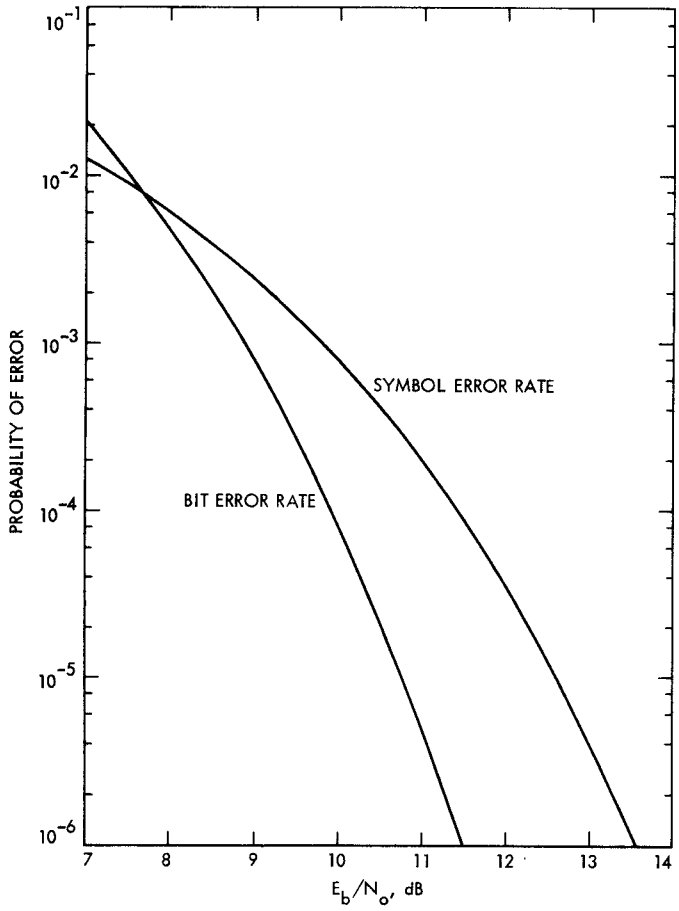

Fig. 2. Performance of quick-look decoder