Node Synchronization in the Block III Maximum-Likelihood Convolutional Decoder

J. M. Layland and J. S. Graham
Communications Ground Systems Section

The Block III maximum-likelihood convolutional decoder (B3MCD) is a programmable convolutional decoder capable of decoding convolutional codes with constraint lengths K from 3 to 15, code rates $1/n$ from $1/2$ to $1/6$, and bit rates as high as 2.2 million bits per second. The B3MCD will acquire node synchronization by examining correlations of reencoded symbols with input soft symbols. Based on simulations, this strategy should achieve node acquisition within 1000 bits with high probability, while robustly detecting error conditions such as all zero bits being transmitted. The acquisition strategy has been verified by simulating the decoder operating on soft symbols recorded from the Block V Receiver in the laboratory. The soft-symbol correlations also will be used to quickly detect loss of synchronization.

I. Introduction

The Block III maximum-likelihood convolutional decoder (B3MCD) is a programmable convolutional decoder capable of decoding convolutional codes with constraint lengths K from 3 to 15, code rates $1/n$ from $1/2$ to $1/6$, and bit rates as high as 2.2 million bits per second.

To correctly decode the incoming symbols, the B3MCD must acquire node synchronization. For a rate $1/n$ code, the decoder generates a bit clock that is n times slower than the incoming symbol clock. The bit clock must be in the correct one of n possible phases to decode the incoming symbols. In addition, while the current DSN (7,1/2) code is transparent (insensitive to inverting all of the input symbols), other codes, such as the (15,1/6) Cassini and Mars Pathfinder codes, are not. In this case, the decoder must also determine the correct polarity of the incoming symbols and correct it before decoding. When synchronizing to the Cassini code, the decoder must correctly choose one of twelve states (six symbol phases and two polarities) to decode the incoming data. Acquisition should be very reliable, yet as quick as possible to lose as little data as possible. The design goal for the decoder asks that, for bit signal-to-noise ratios (SNRs) $E_b/N_0 \geq 0.5$ dB, the decoder should acquire node synchronization within 1000 bit times 50 percent of the time, within 3000 bit times 90 percent of the time, and within 5000 bit times 99 percent of the time; acquisitions should be twice as fast for an E_b/N_0 of 3 dB or more.

While decoding, performance is continually monitored to detect a possible loss of synchronization (polarity or phase). Incorrectly declaring loss of synchronization is an egregious error; the decoder's

requirement asks the decoder not to falsely declare loss of synchronization in 24 hours with a probability of 0.999.

Additionally, we have the practical goals of simplicity, ease of implementation, and robustness to a variety of scenarios. This includes a desire for minimal dependence upon predicted SNR levels. We also would like to detect two (rare) failure scenarios. First, when there is no valid input signal, no node synchronization is possible. Second, when the spacecraft is transmitting long strings of zeros, all possible node synchronizations are equally good.

II. Soft-Symbol Correlation

Previous work on the B3MCD [1] had advocated a combination of three methods for acquiring and monitoring node synchronization. Two of them involve correlating against the known frame markers (either in the bit or symbol domain). Based on results in [2] and [3], either of these methods must integrate over several frame markers (typically four or more, depending on frame marker length and SNR) to acquire node synchronization. With several thousand bits in a frame, this would require a very long time to acquire at low data rates. The requirement stated above asks the decoder to acquire in, at most, a few thousand bits, which would rule out frame marker correlation as the primary acquisition technique.

The third method counted renormalizations in the decoder to estimate the growth rate of the decoder metrics. This technique has the potential to acquire much more quickly. One problem is the requirement to observe several renormalization pulses, setting a minimum time for acquisition. Additional hardware to directly monitor the growth rate of the metrics has been incorporated into the decoder and can be used for acquisition.

The acquisition scheme described here uses a different correlation scheme that does not rely on frame markers. It also will be used for SNR estimation in the decoder. In soft-symbol correlation (Fig. 1), the input soft symbols (appropriately delayed) are correlated against the reencoded bits from the decoder output. This gives almost-known symbols to correlate against the input symbols without waiting for a frame marker. When the decoder is synchronized, the decoded output bits are a very good estimate of the true bits, and hence the reencoded bits should be a good estimate of the true symbols. This will result in a high correlation score. This score can be calculated analytically and will be used by the decoder to estimate the input SNR.

If the decoder is out of synchronization, the decoded bits should be almost random, and the reencoded bits will correlate very poorly with the input symbols. Because of the decoding operation, the correlation will *not* be zero mean. The decoder attempts to output the bit string whose corresponding encoded symbols correlate best with the input, creating a positive bias on the unsynchronized soft-symbol correlation scores. For example, when decoding the (15,1/6) Cassini code, the unsynchronized node phases of the decoder give correlation scores equivalent to an input symbol SNR of about -11 dB, or an E_b/N_0 of about -3 dB.

III. Acquisition Algorithms

To acquire node synchronization, the decoder must test all possible starting offsets for the received symbols. For nontransparent codes, both positive and negative polarities also must be tested. For a standard (7,1/2) code, there are only two possibilities, but for the Cassini (15,1/6) code, there is a total of twelve possibilities to examine. The B3MCD will buffer the input symbols for acquisition and reexamine them in each node position, so that acquisitions with low code rates do not take excessive lengths of time.

The decoder will acquire and monitor node synchronization by examining the soft-symbol correlations of each phase. The expected mean and variance of these correlations can be calculated from the input

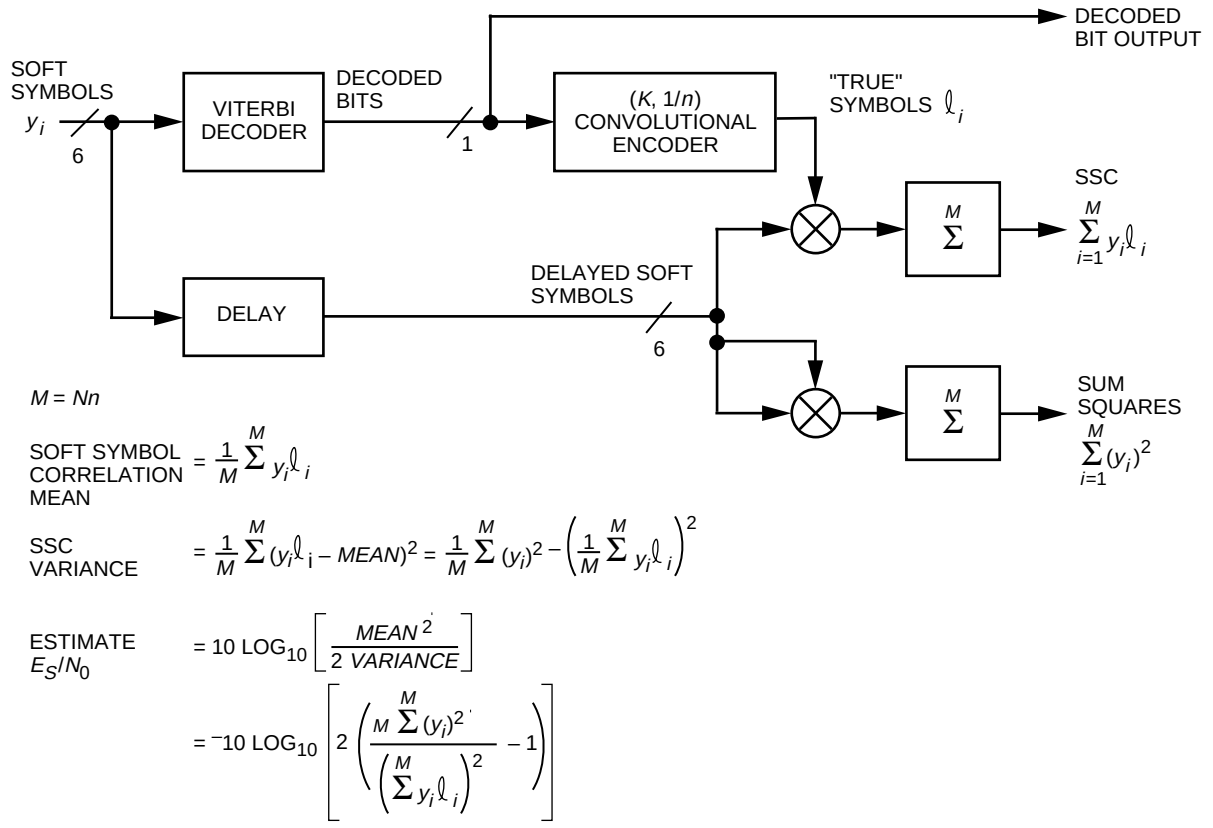

Fig. 1. SNR estimation using soft-symbol correlation.

SNR. Conversely, the symbol SNR can be measured from the mean and variance of the correlations. Specifically, if the standard deviation of the noise on the soft symbols is G, the mean, μ, and standard deviation, σ, of the correlations should be

$$\mu = G\sqrt{2\frac{E_s}{N_0}} \tag{1}$$

$$\sigma = \frac{G}{\sqrt{nN}} \tag{2}$$

Here, N is the number of bits used in the measurement, $1/n$ is the code rate, and E_s/N_0 is the symbol SNR. The correlation mean is equal to the average symbol amplitude, and the correlation variance is the symbol variance reduced by averaging over nN symbols. The parameter G comes from the absolute scaling of the input soft symbols; for the Block V Receiver (BVR), G should be 10 by design (the standard deviation of soft symbols is scaled to 40 in 8 bits, or 10 in 6 bits). As stated previously, the out-of-synchronization positions will have nonzero-mean correlations; these means have not been calculated analytically, but measured by simulation. (Empirically, the out-of-synchronization means are very weakly dependent on SNR, depending almost exclusively on the code being used).

The most straightforward algorithm to analyze sets a threshold based on the predicted SNR. The soft-symbol correlation score is measured in each of the possible phases and polarities. Each of these correlations is compared to the threshold. If exactly one value exceeds the threshold, acquisition is

declared. If not, the process is repeated for a specified number, R, of trials before declaring failure to acquire.

More formally, given parameters N, m, and R, the fixed-threshold algorithm is as follows:

(1) Set the trial counter to zero.

(2) Set the threshold = (predicted correlation score) $- m \times$ (predicted standard deviation).

(3) Measure soft-symbol correlation over N bits in all possible phases and polarities.

(4) Compare each correlation score to the threshold.

(5) If exactly one score exceeds the threshold, declare acquisition.

(6) Otherwise, increment the trial counter. If the counter is $<R$, return to step 3.

(7) Declare failure to acquire.

In addition to finding the correct node position, this algorithm also detects two types of failures. If there are no data at the input, the algorithm should fail to acquire because no correlations exceed the threshold. If the input data are all zeros, so that all phases are equivalent, the acquisition should fail because several (or all) phases are above the threshold.

Some acquisitions in this case may fail even though one of the nodes has a significantly higher correlation than the others. This could occur because the SNR estimate is too high (so the correct node is below the threshold) or too low (so one or more incorrect nodes exceeds the threshold). A much greater problem in practice is the assumption of perfectly known scaling from the receiver. The decoder is relatively insensitive to the scaling of the soft symbols, but this method of acquisition is highly sensitive to the output scaling. This means that the receiver could be supplying decodeable data, yet the decoder would fail to acquire. This sensitivity is eliminated by the next algorithm.

By always choosing the highest node phase, the decoder could acquire node synchronization more accurately with fewer bits, but at the cost of not detecting error conditions. As a result, we tested an acquisition strategy that always selects the highest node position, but sets a threshold dynamically to detect error cases. If we assume the highest node position is accurate, then we can use it to estimate both the variance and the input SNR. This variance is then used to set a threshold as before. We can set the threshold closer to the estimated correlation score than before, because we are no longer concerned that the correct node might fall below the threshold. No-signal cases often will be detected as multiple "high" nodes, because all of the node phases are equivalent (and low). A more robust detection of this state can be made by including an SNR threshold value SNR_{lim}, which the highest node position must exceed to declare acquisition. This value can be set several standard deviations below the lowest decodeable SNR, to minimize the chances of rejecting real data. Given parameters N, m, R, and SNR_{lim}, the complete dynamic threshold algorithm is as follows:

(1) Set the trial counter to zero.

(2) Measure soft-symbol correlation over N bits in all possible phases and polarities.

(3) Measure symbol SNR from the highest correlation position.

(4) If the SNR is $<SNR_{lim}$, go to step 8.

(5) Set the threshold = (highest correlation score) $- m \times$ (standard deviation from this node).

(6) Compare each correlation score to the threshold.

(7) If exactly one score exceeds the threshold, declare acquisition.

(8) Otherwise, increment the trial counter. If the counter is $<R$, return to step 3.

(9) Declare failure to acquire.

Now all thresholds are calculated dynamically from the input data. This eliminates the reliance on SNR predicts and symbol scaling seen in the first algorithm. The SNR_{lim} can be set to the decoding threshold with allowance for measurement inaccuracies.

IV. Acquisition Simulations

A number of simulations of both these algorithms were performed. The idealized input soft symbols were fixed magnitude with additive white Gaussian noise; the standard deviation of the noise was 40 (to match the ideal BVR 8-bit scaling). The results are summarized in Tables 1 through 11. All simulations were run for 1000 test cases. No retries were used in cases of missed detections or multiple high nodes, although their effect can be extrapolated from the results below.

A. Fixed-Threshold Algorithm

The first set of results, presented in Tables 1 through 4, are for the fixed threshold case. The column headings in the tables are as follows:

N = the number of bits used for correlations (not counting 252 start-up bits)

m = the number of standard deviations in the threshold

N_{detect} = the number of times (in 1000) only the correct node exceeded the threshold

N_{miss} = the number of times no nodes exceeded the threshold

N_{multi} = the number of times more than one node exceeded the threshold

N_{false} = the number of times only a wrong node exceeded the threshold (never seen)

These results generally show acquisitions with probabilities of 98 percent or higher using 1000 bits (750 correlation bits plus initialization bits). This probability will be even higher if we try again after a missed acquisition. A few numbers look odd (specifically, the 3-dB cases for the rate-1/2 code in Table 4 are worse using more bits), which presumably are due to statistical fluctuations.

More troubling, however, is the assumption of good SNR predicts used to derive these numbers. Figure 2 shows the effect of relaxing this assumption. The case illustrated is for a 0-dB SNR, $m = 2.5$, rate-1/6 code, allowing the predicts to vary by 1 dB up or down. The value of N_{detect} (as in Table 1) is plotted against the error in the predicted SNR. If this is 0.5 dB, performance is seriously degraded; an error of 1 dB fails completely. Imperfect scaling should cause similar errors.

B. Dynamic Threshold Algorithm

The dynamic threshold method outlined previously attempts to eliminate this reliance on predicted SNR values. There is no analog to Fig. 2 because the predicted SNR is not used in the acquisition. The value of m is now being used only to detect anomalous cases, so a smaller value cannot result in missed acquisitions as in the fixed-threshold algorithm (here the highest node always exceeds the threshold). Based on this, and the results in Tables 5 through 7, a value of $m = 2$ seems plausible.

If we treat the empirical numbers in Tables 5 through 7 as probability estimates, they can be used to calculate overall acquisition probabilities (including retries). Table 8 is for the (15,1/6) code at 0 dB. The columns show probabilities of acquiring on the first, second, or third attempt (assuming two retries are used).

Table 1. Fixed-threshold acquisition, known SNR, for (15,1/6) code at E_b/N_0 = 0 dB.

N	m	N_{detect}	N_{miss}	N_{multi}	N_{false}
1000	2.0	966	34	0	0
1000	2.5	991	9	0	0
1000	3.0	999	1	0	0
1000	3.5	1000	0	0	0
1000	4.0	999	0	1	0
750	2.0	965	35	0	0
750	2.5	986	14	0	0
750	3.0	996	4	0	0
750	3.5	997	1	2	0
750	4.0	988	0	12	0
500	2.0	970	30	0	0
500	2.5	983	14	3	0
500	3.0	973	8	19	0
500	3.5	898	4	98	0
500	4.0	761	1	238	0

Table 2. Fixed threshold acquisition, known SNR, for (15,1/6) code at E_b/N_0 = 3 dB.

N	m	N_{detect}	N_{miss}	N_{multi}	N_{false}
1000	2.5	985	15	0	0
1000	3.0	997	3	0	0
1000	3.5	1000	0	0	0
1000	4.0	1000	0	0	0
500	2.5	982	18	0	0
500	3.0	993	7	0	0
500	3.5	999	1	0	0
500	4.0	999	1	0	0

C. Special Cases

For the special error conditions mentioned previously, simulations were run at 1.5 dB for the (7,1/2) code and at 0 dB for the (15,1/6) code. With all zeros transmitted, in both cases all 1000 simulations detected multiple high nodes.

For the case with no data at the input, when the receiver expects 0-dB (15,1/6) data, the decoder rejected all 1000 test cases when the threshold was set at −8.72 dB (three standard deviations below an −8.0-dB symbol SNR, or a −0.2-dB bit SNR). When the threshold was set even lower at −9.52 (three standard deviations below a −1-dB bit SNR), the algorithm rejected 932 cases (some because of multiple "high" nodes).

Table 3. Fixed threshold acquisition, known SNR, for (7,1/2) code at E_b/N_0 = 1.5 dB.

N	m	N_{detect}	N_{miss}	N_{multi}	N_{false}
500	2.0	947	41	12	0
500	2.5	937	10	53	0
500	3.0	778	3	219	0
500	3.5	536	0	464	0
750	2.0	942	58	0	0
750	2.5	985	11	4	0
750	3.0	973	1	26	0
750	3.5	888	0	112	0
1000	2.0	951	49	0	0
1000	2.5	984	15	1	0
1000	3.0	997	2	1	0
1000	3.5	982	1	17	0
1000	4.0	931	0	69	0

Table 4. Fixed threshold acquisition, known SNR, for (7,1/2) code at E_b/N_0 = 3 dB.

N	m	N_{detect}	N_{miss}	N_{multi}	N_{false}
500	2.0	904	96	0	0
500	2.5	970	30	0	0
500	3.0	992	8	0	0
500	3.5	997	3	0	0
500	4.0	999	1	0	0
750	2.0	888	112	0	0
750	2.5	964	36	0	0
750	3.0	995	5	0	0
750	3.5	1000	0	0	0
1000	4.0	1000	0	0	0
1000	2.0	870	130	0	0
1000	2.5	951	49	0	0
1000	3.0	985	15	0	0
1000	3.5	997	3	0	0
1000	4.0	998	2	0	0

D. Laboratory Tests

The dynamic threshold acquisition algorithm was tested on soft-symbol data recorded by the BVR using test software tracking a test signal in the Telecommunications Data Laboratory (TDL). The test bits were taken from a 2047-state pseudorandom bit generator. The symbol rate was 600 symbols/second in all cases (e.g., 100 bits/second for the rate-1/6 code). There was a problem in the connection used

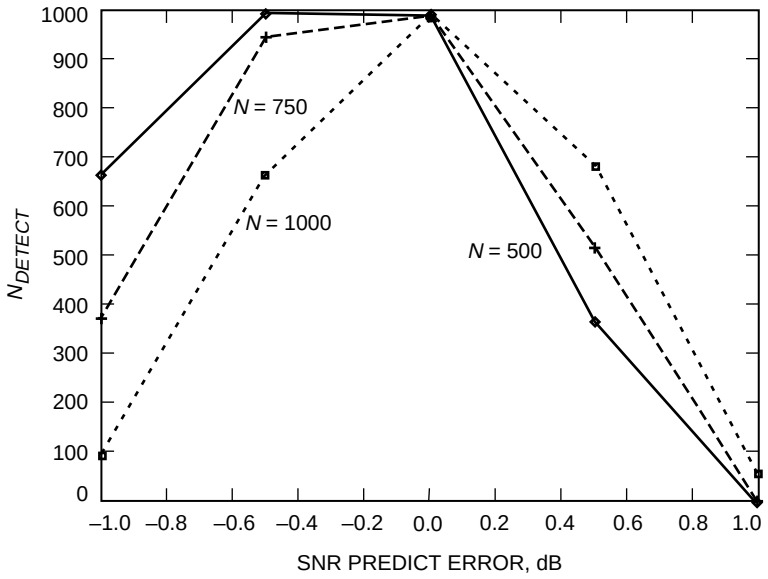

Fig 2. The effect of predict errors on the fixed-threshold acquisition algorithm.

Table 5. Acquisition results for dynamic threshold, (15,1/6) code at E_b/N_0 = 0 dB.

N	m	N_{detect}	N_{miss}	N_{multi}	N_{false}
1000	2.0	1000	0	0	0
1000	2.5	1000	0	0	0
1000	3.0	1000	0	0	0
1000	3.5	1000	0	0	0
1000	4.0	1000	0	0	0
750	2.0	1000	0	0	0
750	2.5	999	0	1	0
750	3.0	996	0	4	0
750	3.5	984	0	16	0
750	4.0	961	0	39	0
500	2.0	981	0	19	0
500	2.5	956	0	44	0
500	3.0	902	0	98	0
500	3.5	812	0	188	0
500	4.0	676	0	324	0

to record the soft symbols, which resulted in periodically dropping one second's worth of data. (This is a known problem with the test software used for the recording.) Because of the integral bit rate used, these dropouts never resulted in a loss of synchronization. Since there appeared to be little effect on the results, no effort was made to exclude these gaps from the data used.

Tests were run at E_b/N_0's of 2 and 0 dB for the (15,1/6) Pathfinder code, and at 3 and 1.5 dB using the DSN standard (7,1/2) code. The acquisitions used 750 correlation bits (1002 bits total) with $m = 2$. All acquisitions were successful; the results are summarized in Table 9.

Table 6. Acquisition results for dynamic threshold, (15,1/6) code at E_b/N_0 = 3 dB.

N	m	N_{detect}	N_{miss}	N_{multi}	N_{false}
1000	2.0	1000	0	0	0
1000	2.5	1000	0	0	0
1000	3.0	1000	0	0	0
1000	3.5	1000	0	0	0
1000	4.0	1000	0	0	0
500	2.0	1000	0	0	0
500	2.5	1000	0	0	0
500	3.0	1000	0	0	0
500	3.5	1000	0	0	0
500	4.0	1000	0	0	0

Table 7. Acquisition results for dynamic threshold, (7,1/2) code at E_b/N_0 = 1.5 dB.

N	m	N_{detect}	N_{miss}	N_{multi}	N_{false}
750	2.0	1000	0	0	0
750	2.5	994	0	6	0
750	3.0	958	0	42	0
500	2.0	987	0	13	0
500	2.5	912	0	88	0
500	3.0	672	0	328	0

Table 8. Acquisition probabilities for (15,1/6) code at E_b/N_0 = 0 dB, using retries.

N	m	First, percent	Second, percent	Third, percent
500	2.0	98.1	99.96	99.999
500	3.0	90.2	99.04	99.906
750	2.0	100.0	100.0	100.0
750	3.0	99.6	99.998	100.0
1000	2.0	100.0	100.0	100.0
1000	3.0	100.0	100.0	100.0

Table 9. Acquisition results from laboratory tests.

Code	E_b/N_o, dB	Acquisitions	Comments
(7,1/2)	2.9	1168/1168	No retries used.
(7,1/2)	1.4	1622/1622	Two acquisitions required one retry.
(15,1/6)	0.0	1065/1065	No retries used.
(15,1/6)	2.0	1332/1332	No retries used.

The fixed-threshold acquisition algorithm also was run on these data sets, with abysmal results. As stated previously, this algorithm is highly sensitive to the exact scaling of the soft symbols. This caused no problems using perfect scaling in the simulations reported previously, but the actual variations (up to a factor of two) seen in the laboratory were large enough that the fixed threshold virtually never worked.

V. Loss of Synchronization Detection

While decoding, the decoder must continually monitor its performance to detect loss of synchronization. As in the node acquisition studies above, the original intention was to detect loss of synchronization by monitoring soft-symbol correlation (SSC) and comparing against a threshold. Ideally, this threshold should depend solely on the code being used, and not on the SNR. Unfortunately, an inspection of the simulation results shows this to be an impossibility. At high SNRs, both the in- and out-of-synchronization correlation scores rise. This is because the decoder can find some partial correlations even when not synchronized, and the correlation score scales with average symbol value, which increases with SNR. Table 10 shows this effect, and suggests an alternate approach.

The soft-symbol correlation numbers in Table 10 show that unsynchronized cases at high SNRs exceed the average correlation scores of synchronized cases at lower SNRs. Therefore, any threshold usable at low SNRs would be unable to detect loss of node synchronization at high SNRs. Longer integration times would tighten the standard deviations, but would not change the means. While 20 dB was included in Table 10 as an "infinite SNR" limit, this effect exists even at 3 dB for the rate-1/2 code.

The last two columns in Table 10 show the decoder's SNR estimate (in dB). This estimate uses both the correlation score (to get mean symbol value) and an estimate of the variance. As can be seen in the table, unsynchronized cases still give low SNR estimates even at high E_b/N_0. Even at unrealistically high SNRs, the symbol SNR estimate is still below the decoding threshold. The SNR estimate is not affected

Table 10. Soft-symbol correlation and SNR estimate statistics dependence on input SNR.

Code	E_b/N_0, dB	N	Estimated SSC (in)	Estimated SSC (out)	Estimated SNR (in), dB	Estimated SNR (out), dB
(7,1/2)	1.5	3000	11.79 ± 0.12	10.77 ± 0.12	-1.35 ± 0.11	$-\ 3.08 \pm 0.08$
(7,1/2)	3.0	3000	13.93 ± 0.12	12.06 ± 0.10	0.22 ± 0.11	$-\ 2.89 \pm 0.08$
(7,1/2)	20	3000	30.99 ± 0.00	23.20 ± 0.08	$31.8\ \pm 0.00$	$-\ 1.96 \pm 0.07$
(15,1/6)	0.0	1000	5.75 ± 0.12	4.38 ± 0.27	-7.77 ± 0.20	-10.72 ± 0.62
(15,1/6)	3.0	1000	8.10 ± 0.12	4.89 ± 0.29	-4.75 ± 0.15	-10.68 ± 0.60
(15,1/6)	20	1000	30.96 ± 0.00	12.27 ± 0.70	$26.7\ \pm 0.2$	-10.32 ± 0.60

by the increasing symbol magnitude, because the variance estimate increases also. The SNR estimator is known to be biased at high SNRs, so the 20-dB SNR estimates are higher than reality. The 0.2-dB upward bias on the rate-1/2 SNR estimates could be due to the same effect that causes no-signal cases to look like low SNRs after decoding.

Due to the large penalty in lost data caused by relocking the telemetry system, the decoder should be *very* certain before declaring loss of synchronization. Specifically, a requirement states that this should not happen in 24 hours at 2.2 Mbps with a probability of 0.999. Alternatively, the decoder should falsely declare loss of synchronization at most once in 1000 days of continuous operation. If we make a synchronization decision at most once per second, then the requirement translates into 1 false decision out of 86,400,000, or roughly $P_{FA} = 10^{-8}$. Assuming Gaussian statistics, this means the threshold should be set at least 5.6 standard deviations below the expected mean. Since a loss of synchronization detection will require human intervention to diagnose and correct the problem (possibly by reacquiring), the 1-second assumption seems reasonable even for high data rates.

Using the correlation SNR estimate as our decision variable, and setting a threshold at 5.6 standard deviations below the mean SNR estimate at 1.5 dB (rate-1/2) or 0 dB (rate-1/6), we can use Gaussian statistics to estimate the probability of detecting loss of synchronization. Table 11 shows that 1000 bits are enough to give a high probability of detecting loss of synchronization. Note that the last column is the probability of *not* detecting loss of node synchronization, so low numbers are desirable. Obviously, at high data rates, many more bits are available every second and would be used. Using more bits in the estimate does not affect the threshold, which can remain fixed for each code, but will tighten the standard deviations and result in even higher detection probabilities. For example, the last two rows show that the probability of failing to detect loss of node synchronization in 3000 bits is infinitessimal.

Table 11. Probability of detecting loss of synchronization in *N* bits.

Code	E_b/N_0, dB	N	$1 - P_d$	Threshold, dB
(15,1/6)	0	1000	1.6×10^{-3}	−8.9
(15,1/6)	3	1000	1.4×10^{-3}	−8.9
(7,1/2)	1.5	1000	2.0×10^{-5}	−2.5
(7,1/2)	3	1000	3.9×10^{-3}	−2.5
(7,1/2)	1.5	3000	2×10^{-44}	−2.0
(7,1/2)	3	3000	4×10^{-31}	−2.0

The numbers in Tables 10 and 11 were calculated based on statistics from the simulations. While it is straightforward to estimate the statistics of the correlations when the decoder is in synchronization, it is not clear how to estimate these numbers for the out-of-synchronization cases. The problem is that the decoder tries to find the best match among possible symbol sequences and creates correlations where there should be none. For example, Table 10 showed that unsynchronized (7,1/2) symbols look like a −3 dB symbol SNR to the decoder.

The actual method by which the probabilities were calculated is as follows: First, the threshold was set at 5.6 standard deviations below the mean SNR at the lowest SNR used. These numbers were taken from simulation, although they could be calculated approximately (and obviously the mean should be the actual symbol SNR). Next, the out-of-synchronization mean (from simulation) was subtracted from the threshold. This margin was divided by the out-of-synchronization standard deviation (again from simulation) to determine a margin in normalized units. Finally, the Gaussian distribution was used to

determine the probability of an out-of-synchronization case exceeding a threshold this number of standard deviations above its mean.

In actual practice, determining thresholds from simulations should not be a problem, since we need only one threshold per code. The thresholds as computed above can be estimated analytically, but simulation takes into account all effects of the decoder on the SNR estimation. Simulations are still necessary to determine the probability of detecting loss of synchronization, unless a method exists to estimate the out-of-synchronization statistics.

VI. Current Status and Future Work

B3MCD hardware has been fabricated and currently is being tested. This is still very preliminary, including some software development and debugging, but acquisitions appear to take approximately 1000 bits as designed. A 64-chip B3MCD board decoding a (15,1/6) code has passed compatibility tests with Mars Pathfinder.[1] Detailed statistical characterization of acquisition times to verify these results remains to be done.

The acquisition scheme described here was designed to be as robust as possible. Some early studies suggest that reliable acquisitions can be made with many fewer bits (a few hundred). These trade-offs between speed and robustness remain to be characterized.

Acknowledgments

The authors thank the Communications Systems and Research Section for providing the original simulation code used in this work, and Ramona Tung for modifying it to match the B3MCD design. Kar-Ming Cheung provided valuable discussions and advice throughout this project. We also thank Bob Johnson, for his continuing support, and the entire B3MCD team for its hard work in bringing this design to completion.

References

[1] J. I. Statman, B. Siev, and J. Rabkin, "Node and Frame Synchronization in the Big Viterbi Decoder," *The Telecommunications and Data Acquisition Progress Report 42-103, July–September 1990*, Jet Propulsion Laboratory, Pasadena, California, pp. 154–157, November 15, 1990.

[2] K.-M. Cheung and J. I. Statman, "A Node and Frame Synchronization in the Big Viterbi Decoder Based on Channel Symbol Measurements," *The Telecommunications and Data Acquisition Progress Report 42-103, July–September 1990*, Jet Propulsion Laboratory, Pasadena, California, pp. 203–214, November 15, 1990.

[3] K.-M. Cheung and J. I. Statman, "Node Synchronization Schemes for the Big Viterbi Decoder," *The Telecommunications and Data Acquisition Progress Report 42-108, October–December 1992*, Jet Propulsion Laboratory, Pasadena, California, pp. 186–200, February 15, 1992.

[1] L. Harcke, "Mars Pathfinder (15,1/6) Convolutional Code Compatibility and Performance Test Report," JPL Interoffice Memorandum 3312-96-047 (internal document), Jet Propulsion Laboratory, Pasadena, California, November 19, 1996.