Enhanced Decoding for the Galileo Low-Gain Antenna Mission: Viterbi Redecoding With Four Decoding Stages

S. Dolinar and M. Belongie
Communications Systems Research Section

The Galileo low-gain antenna mission will be supported by a coding system that uses a (14,1/4) inner convolutional code concatenated with Reed–Solomon codes of four different redundancies. Decoding for this code is designed to proceed in four distinct stages of Viterbi decoding followed by Reed–Solomon decoding. In each successive stage, the Reed–Solomon decoder only tries to decode the highest redundancy codewords not yet decoded in previous stages, and the Viterbi decoder redecodes its data utilizing the known symbols from previously decoded Reed–Solomon codewords.

A previous article [1] analyzed a two-stage decoding option that was not selected by Galileo. The present article analyzes the four-stage decoding scheme and derives the near-optimum set of redundancies selected for use by Galileo. The performance improvements relative to one- and two-stage decoding systems are evaluated.

I. Introduction

This article is a follow-on to [1], which analyzed two enhanced decoding options planned for the Galileo low-gain antenna (LGA) mission: Reed–Solomon redecoding using erasure declarations and Viterbi redecoding using Reed–Solomon corrected symbols. The analysis in [1] produced tables of gains achievable from enhanced decoding under an assumption of infinite interleaving for one, two, or four stages of Viterbi decoding, but no Reed–Solomon redecoding, and for one or two stages of Viterbi decoding, with or without Reed-Solomon redecoding, under the actual Galileo conditions of depth-8 interleaving. The present article looks at the case of four stages of Viterbi decoding and depth-8 interleaving. The four-stage coding system has been selected for implementation to support the Galileo LGA mission.

II. Block Diagram of Coding Options

A block diagram of the various coding options is shown in Fig. 1. A Reed–Solomon encoded data block is interleaved to depth 8 and then encoded by the (14,1/4) convolutional encoder. The Reed–Solomon codewords can have four different levels of redundancies, as depicted by the lightly shaded areas at the bottom of the code block in Fig. 1. The encoded data are modulated, passed over an additive white Gaussian noise (AWGN) channel, demodulated, and presented to a Viterbi decoder. After

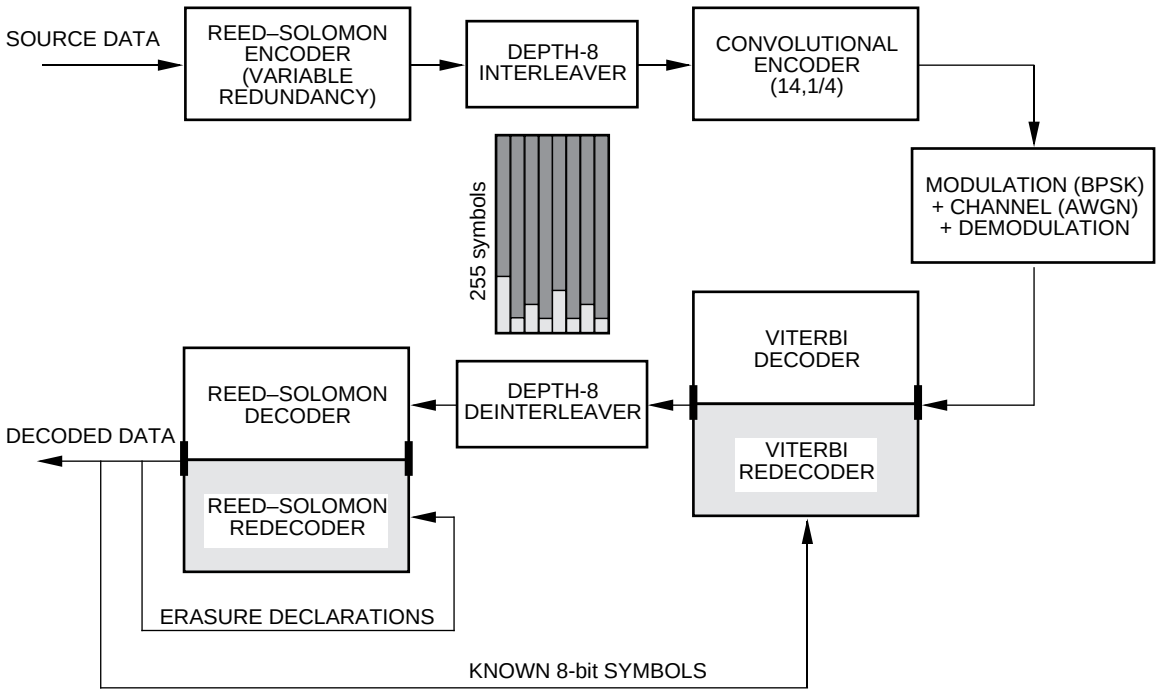

Fig. 1. Coding options.

deinterleaving, the codeword or set of codewords with the highest redundancy is decoded by the Reed–Solomon decoder. The symbols in the codeword(s) decoded by the Reed–Solomon decoder are fed back to assist the Viterbi decoder in *redecoding* the symbols in weaker codewords. The output of the Viterbi redecoder is deinterleaved, and the set of codewords with the next highest redundancy is then decoded by the Reed–Solomon decoder. The newly decoded symbols are fed back to further assist the Viterbi redecoder, and the process is repeated for two more decoding stages until the codewords in all four redundancy classes are successfully decoded.

Figure 1 also shows an option for a shorter feedback loop entirely within the Reed–Solomon decoder using erasure declarations. As shown in [1], Reed–Solomon redecoding using erasure declarations based on error forecasting was worth around 0.19 dB when used in conjunction with one-stage decoding of the Galileo LGA convolutional code. However, the extra gain from using erasure declarations shrinks to a minuscule 0.02 dB when combined with two-stage Viterbi decoding. For four-stage decoding, the marginal improvements gained from erasure declarations are almost nil. Therefore, in the present article, Reed–Solomon redecoding using erasure declarations has not been considered in analyzing four-stage decoding performance. However, the Galileo LGA coding system will still incorporate the capability to perform this type of redecoding, as it may prove helpful in overcoming decoding difficulties not caused by AWGN, such as closing data gaps caused by unsynchronized symbols.

III. The Simulation Data

Figures 2 through 5 are improved and expanded versions of Figs. 1 through 4 of [1], obtained by accumulating many more millions and billions of simulated decoded bits during the interim. Figure 2 shows the bit error rate (BER) and symbol error rate (SER) (for 8-bit Reed–Solomon symbols) for convolutionally encoded symbols decoded by either the Big Viterbi Decoder (BVD) or a software (S/W) Viterbi decoder. The software decoding algorithm is a close approximation to the software decoder that is actually being designed to support the Galileo LGA mission. Figure 3 shows the decoded symbol error

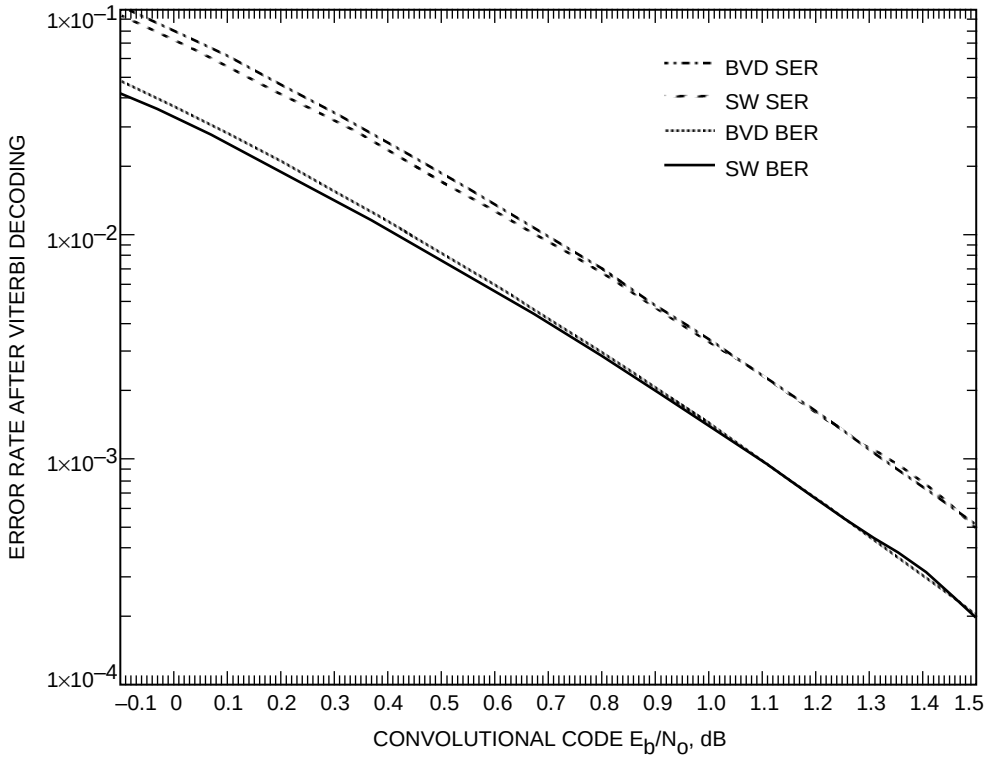

Fig. 2. BER and SER after Viterbi decoding of the (14,1/4) Galileo LGA convolutional code by the BVD and by a software decoder.

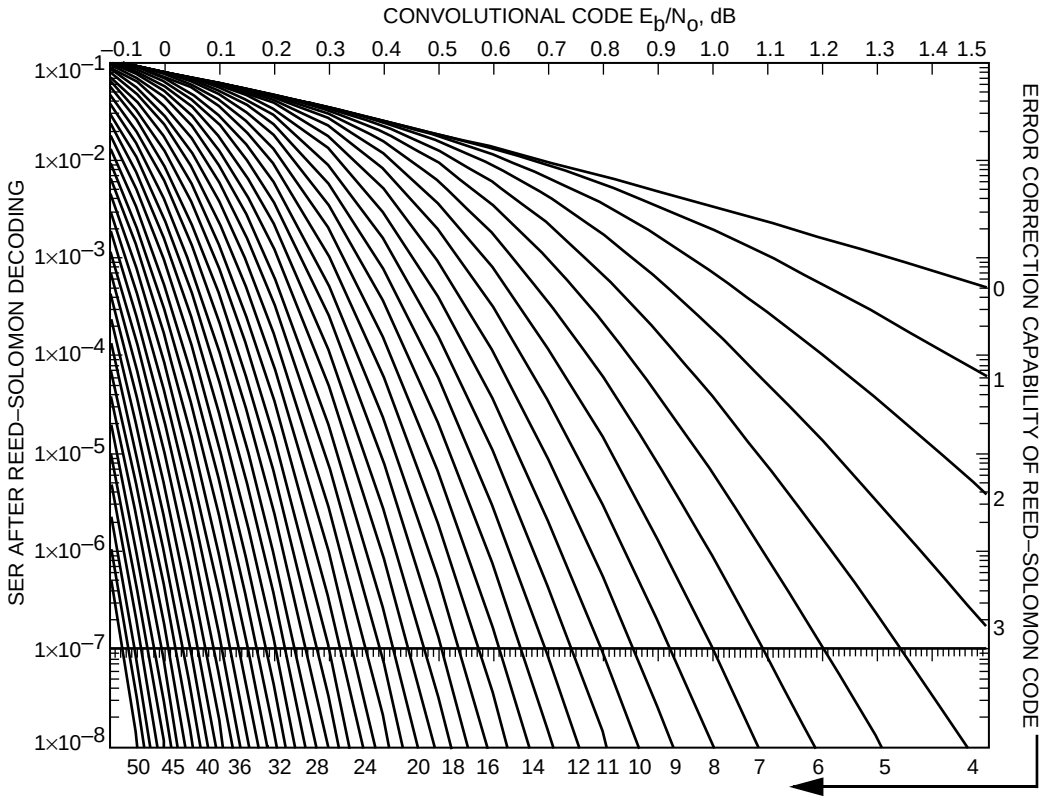

Fig. 3. SER after decoding inner convolutional code with BVD followed by Reed–Solomon decoding with infinite interleaving.

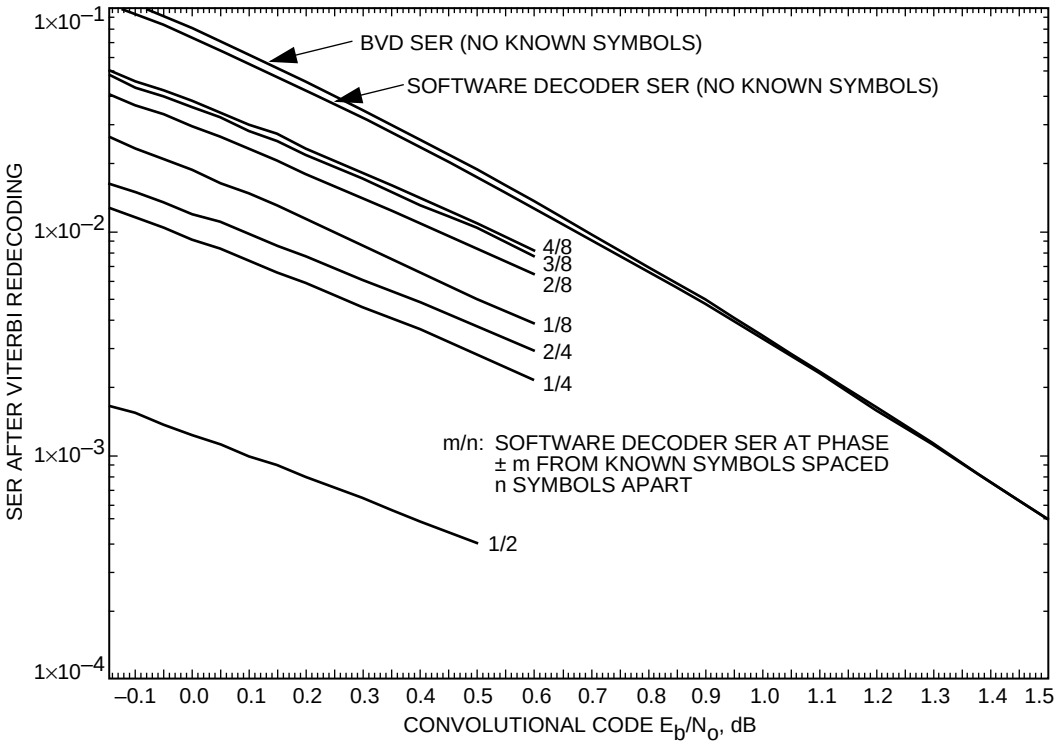

Fig. 4. SER after Viterbi redecoding with the software decoder for various spacings of known symbols and relative phases of unknown symbols.

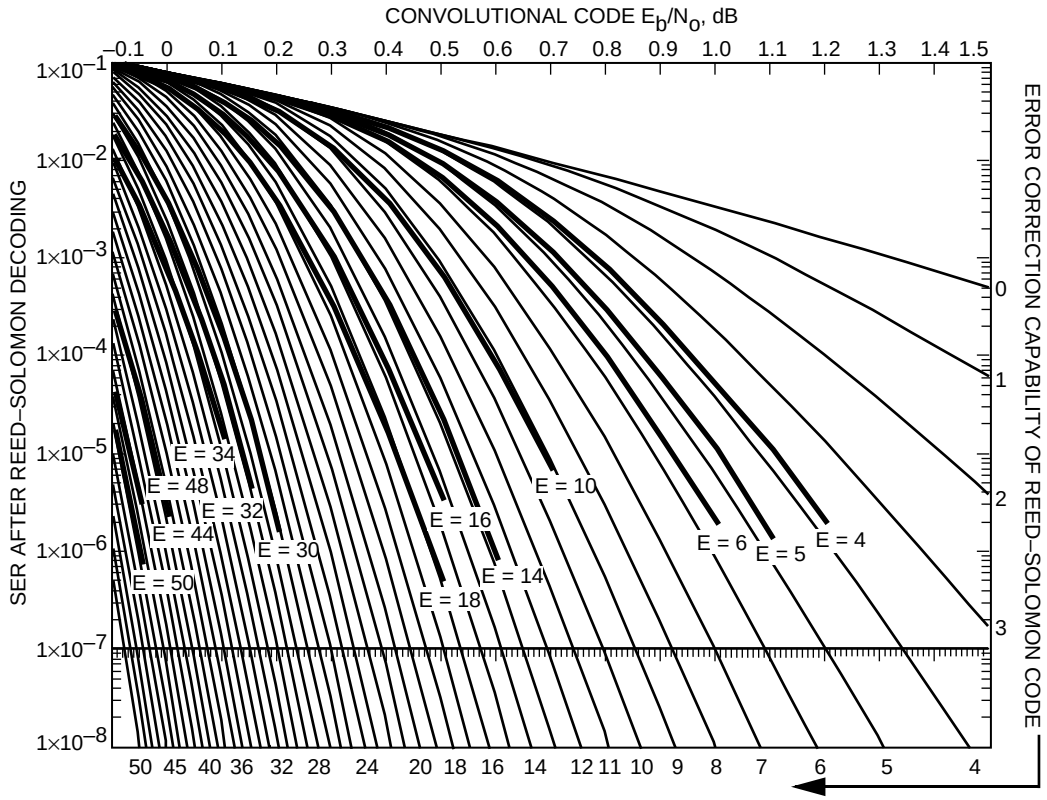

Fig. 5. SER after decoding inner convolutional code with BVD followed by Reed–Solomon decoding with depth-8 interleaving.

rate for a Reed–Solomon decoder receiving convolutionally decoded bits from the BVD; the x-axis of Fig. 3 is the convolutional code signal-to-noise ratio (SNR) E_b/N_o. Figure 4 shows the decoder symbol error rate for the software Viterbi decoder presented with known symbols repeating once every eight, four, or two symbols; as discussed in [1], these SERs depend on the phase of the decoded symbols relative to the locations of the known symbols. The baseline SER curves from Fig. 2 for no known symbols are also included in this figure for reference. Figure 5 repeats the infinite interleaving performance curves from Fig. 3 and overlays curves representing Reed–Solomon decoded SER when the codewords are interleaved to depth 8. As in Fig. 3, the Reed–Solomon decoder for Fig. 5 receives its symbols from the output of the BVD, and the x-axis measures the signal-to-noise ratio at the output of the BVD, not the overall signal-to-noise ratio at the output of the concatenated Reed–Solomon and convolutional codes. SER estimates in Figs. 2 through 5 were taken at spacings of 0.05 or 0.10 dB, and each estimate was based on about 2 Gbits of decoded data from the BVD or 25 to 100 Mbits of data from the software decoder.

As is evident from Figs. 2 and 4, the software decoder performs a few hundredths of a dB better than the BVD (due to a longer truncation length and other factors). The analysis of four-stage decoding requires the use of both Figs. 4 and 5; proper calibration is important between the software-decoder-based curves in Fig. 4 and BVD-based curves in Figure 5. In [1], no distinction was made between the performance of the two decoders, because the software decoder at that time resembled the BVD more closely than the ultimate Galileo LGA decoder. From Fig. 4 is deduced a table of SER-equivalent E_b/N_o operating points for the BVD operating with no known symbols. Whenever the software decoder is decoding data at a value of E_b/N_o in the leftmost column of Table 1, the BVD achieves the same average SER at the "equivalent" E_b/N_o in the columns to the right. There is one BVD-equivalent E_b/N_o column for each of the software-decoder-based curves in Fig. 4. For the case of no known symbols, this really is a near equivalence, and the decoded bit errors from the software decoder and the BVD have very similar burst statistics, not just average SER. For the various cases of known symbols presented to the software decoder, this equivalence is only in terms of average SER. As noted in [1], the error bursts from a decoder presented with known symbols are more benign than those for a decoder operating at the same average SER without any known symbols, as measured by their effects on Reed–Solomon decoding with finite interleaving. Thus, use of the BVD-equivalent signal-to-noise ratios in Table 1 will give slightly conservative predictions of performance in decoding stages 2 through 4.

Table 1. BVD-equivalent signal-to-noise ratios E_b/N_o, dB.

Software decoder E_b/N_o, dB	Known symbol phase/spacing input to software decoder							
	None	4/8	3/8	2/8	1/8	2/4	1/4	1/2
−0.15	—	0.16	0.18	0.24	0.39	0.54	0.62	1.20
−0.10	−0.06	0.20	0.22	0.28	0.43	0.57	0.65	1.22
−0.05	−0.01	0.23	0.25	0.32	0.46	0.61	0.69	1.25
0.00	0.04	0.27	0.29	0.36	0.50	0.64	0.72	1.28
0.05	0.09	0.31	0.33	0.39	0.54	0.67	0.75	1.30
0.10	0.14	0.35	0.37	0.43	0.58	0.70	0.78	1.33
0.15	0.18	0.38	0.41	0.47	0.61	0.74	0.82	1.36
0.20	0.23	0.43	0.45	0.51	0.66	0.77	0.85	1.39
0.30	0.33	0.51	0.53	0.59	0.74	0.84	0.92	1.44
0.40	0.43	0.59	0.61	0.67	0.82	0.91	0.99	—
0.50	0.53	0.67	0.69	0.75	0.90	0.98	1.06	—
0.60	0.62	0.75	0.77	0.82	0.97	1.04	1.13	—

IV. The Basic Analysis Procedure

As noted in [1], even 2 Gbits of BVD-decoded data are insufficient to directly verify Reed–Solomon decoded SERs around 10^{-7} for the case of depth-8 interleaving. Instead, such performance must be inferred by extrapolating the simulated depth-8 curves along the accurately known family of curves for infinite interleaving. Each curve for depth-8 interleaving becomes nearly parallel to a member of the family of infinite interleaving curves, and 10^{-7} performance for depth-8 interleaving may be inferred by extrapolating along an "equivalent" infinite interleaving curve.

The selection and analysis of an appropriate set of codeword redundancies for four-stage decoding is illustrated in the following example. First, select a desired E_b/N_o operating point for the inner convolutional code using the software decoder. This choice is somewhat arbitrary, because the same analysis must be repeated for several values of E_b/N_o in order to determine the optimum operating point. For this example, a convolutional code signal-to-noise ratio E_b/N_o of 0.00 dB will be used. From Table 1, the average SER from the first stage of Viterbi decoding by the software decoder is the same as the average SER produced by the BVD at the BVD-equivalent operating point of 0.04 dB. The output SER from the first Reed–Solomon decoder stage is obtained from the BVD's performance curve in Fig. 5. If the target SER is around 2×10^{-7} (target BER around 1×10^{-7}), the highest redundancy codewords must yield an output SER on the order of 10^{-7} without any help from succeeding decoding stages. From Fig. 5, this can be accomplished at a BVD-equivalent signal-to-noise ratio E_b/N_o of 0.04 dB by using a codeword with correction capability E of approximately 47. From Table 1, the average SER from the second stage of Viterbi decoding with one known symbol every eight is the same as the average SER produced by the BVD with no known symbols at the BVD-equivalent operating points of 0.50, 0.36, 0.29, and 0.27 dB, for codewords with symbols at phases ± 1, ± 2, ± 3, and ± 4, respectively, from the known symbol. From Fig. 5, codewords with $E \approx 20$, 26, 29, and 30, respectively, can achieve SERs just under 10^{-7} for these four phases. Looking ahead to the next stage of Viterbi decoding, it can be shown that the biggest payoff comes from locating the second highest redundancy codeword at phase ± 4. Then the third stage of Viterbi decoding is accomplished with one known symbol every four, and the BVD-equivalent operating points from Table 1 are 0.72 and 0.64 dB for phases ± 1 and ± 2, respectively. These require codewords with $E \approx 13$ and 15, respectively, and again it can be shown that the out-of-phase location ± 2 makes the best utilization of the fourth and final Viterbi decoding stage. With two of these third highest redundancy codewords per block of eight placed at phases ± 2, the final Viterbi decoding operation is accomplished with one known symbol every two, and from Table 1, the BVD-equivalent operating point for the unknown symbols at phase ± 1 is 1.28 dB, requiring four lowest-redundancy codewords with $E \approx 5$. This selection process yields a redundancy profile $2E \approx (94, 10, 30, 10, 60, 10, 30, 10)$; this incurs a redundancy overhead cost of 0.58 dB, and the resulting concatenated code signal-to-noise ratio E_b/N_o is 0.58 dB. The overall average SER achieved by four-stage decoding using this redundancy set can be computed approximately by the formula given in [1], $SER = SER_a(1) + 7/8\ SER_b(2) + 3/4\ SER_c(3) + 1/2\ SER_d(4)$, where the indices a, b, c, and d refer to the strongest, next strongest, third strongest, and weakest codewords and (n) refers to decoding during stage n, $n = 1, 2, 3, 4$. Extrapolations from Fig. 5 for $SER_a(1)$, $SER_b(2)$, $SER_c(3)$, and $SER_d(4)$ yield an overall SER of approximately 2×10^{-7}.

Similar analyses starting with convolutional code operating points different from 0.00 dB yield different sets of optimal redundancies and different concatenated code E_b/N_o. It can be shown empirically that the optimum convolutional code operating point for four-stage decoding occurs within a range from approximately -0.10 to $+0.05$ dB, and that essentially identical performance (within one or two hundredths of a dB) is achievable by suitably selecting different redundancy sets within this range. Also, the best pattern of codeword redundancies always appears to be (a, d, c, d, b, d, c, d), where a is the highest redundancy, b the next highest, c the third highest, and d the lowest. This is the same pattern suggested by an earlier analysis of four-stage decoding in [2].

V. A More Refined Analysis Procedure

The analysis above may be refined by further studying the relationship between the performance curves for depth-8 interleaving and the "equivalent" infinite interleaving curves along which depth-8 SERs on the order of 10^{-7} are extrapolated. Table 2 and Fig. 6 attempt to quantify this equivalence.

Table 2. Equivalent error correction needed for infinite interleaving to yield the same SER.

BVD E_b/N_o, dB	Error correction for depth-8 interleaving												
	$E=4$	$E=5$	$E=6$	$E=10$	$E=14$	$E=16$	$E=18$	$E=30$	$E=32$	$E=34$	$E=44$	$E=48$	$E=50$
−0.10	—	—	—	—	—	—	—	29.16	30.82	32.46	40.60	43.77	45.18
−0.05	—	—	—	—	—	—	—	28.70	30.34	31.98	40.07	43.44	—
0.00	—	—	—	—	—	—	—	28.26	29.89	31.50	39.90	—	—
0.05	—	—	—	—	—	—	—	27.86	29.45	31.08	—	—	—
0.10	—	—	—	—	—	—	—	27.51	29.10	—	—	—	—
0.15	—	—	—	—	13.78	15.52	17.24	27.25	29.15	—	—	—	—
0.20	—	—	—	—	13.57	15.30	17.02	—	—	—	—	—	—
0.30	—	—	—	9.73	13.23	14.94	16.63	—	—	—	—	—	—
0.40	—	—	—	9.52	12.99	14.74	16.34	—	—	—	—	—	—
0.50	3.96	4.89	5.80	9.36	12.86	14.49	—	—	—	—	—	—	—
0.60	3.88	4.82	5.72	9.31	—	—	—	—	—	—	—	—	—
0.70	3.82	4.73	5.64	9.17	—	—	—	—	—	—	—	—	—
0.80	3.81	4.73	5.63	—	—	—	—	—	—	—	—	—	—
0.90	3.81	4.70	5.64	—	—	—	—	—	—	—	—	—	—
1.00	3.84	4.67	5.59	—	—	—	—	—	—	—	—	—	—
1.10	3.79	4.76	—	—	—	—	—	—	—	—	—	—	—

Table 2 shows, for each Reed–Solomon code tested at depth-8 interleaving, the equivalent error correction capability needed to achieve the same SER if the interleaving were ideal. At each value of E_b/N_o, the equivalent error correction is obtained by linear interpolation on the log scale between the two adjacent infinite interleaving curves. It is quoted as a real number, not an integer, and thus does not represent a realizable code. For example, from Fig. 5 at 0.5 dB, the $E = 16$ curve for depth-8 achieves an SER about halfway between the infinite interleaving curves for $E = 14$ and $E = 15$. The corresponding equivalent error correction capability is listed in Table 2 as $E = 14.49$.

Figure 6 plots a normalized version of the numbers in Table 2. Each point in Table 2 is plotted with an x-coordinate equal to the depth-8 SER at the given value of E_b/N_o and a y-coordinate equal to the ratio of the actual depth-8 error correction capability to the equivalent infinite interleaving error-correction capability listed in Table 2. This ratio is referred to as the depth-8 error magnification factor. For purposes of computing Reed–Solomon code performance, the (nonindependent) symbol errors occurring in depth-8 interleaved codewords are effectively multiplied by the error magnification factor, as compared to an equal average number of independent symbol errors. The error magnification factor is a way of measuring the propensity for one long Viterbi decoder error burst to contribute more than one symbol error to a given Reed–Solomon codeword whenever the codewords are only finitely interleaved.

A more mechanized approach than visually extrapolating the depth-8 performance curves in Fig. 5 utilizes the error magnification factors presented in Fig. 6. The first step is to solve for the redundancies that

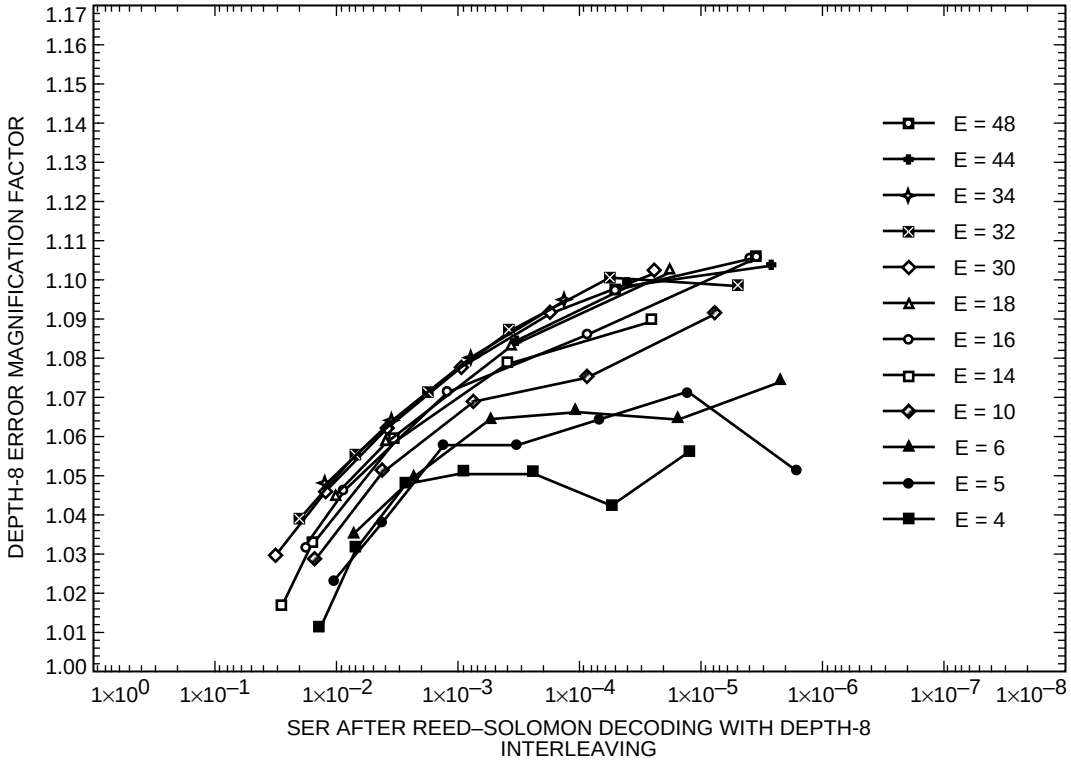

Fig. 6. Effective error magnification factors for Reed–Solomon decoding with depth-8 interleaving, as compared to Reed–Solomon decoding with infinite interleaving.

would be needed if the interleaving were ideal. For this solution, fractional redundancies and fractional error correction capabilities are permissible, and these can be obtained very accurately by interpolation between successive ideal interleaving curves. At each convolutional code operating point, the goal is to solve for a roughly "balanced" set of four redundancies, a, b, c, d, used in the pattern (a, d, c, d, b, d, c, d). A balanced set of redundancies is one for which each class of codewords contributes roughly equally to the overall SER. If the redundancies were not roughly balanced, essentially the same performance could be achieved at lower cost by reducing the redundancy of a codeword class that contributes only a tiny portion of the overall SER.

After a balanced set of fractional redundancies for ideal interleaving is obtained, the next step is to scale these upward by the error magnification factors for depth-8 interleaving and then round these numbers to the nearest or next higher even-integer $2E$. The integer roundoff causes some loss of balance and could cause worse performance if all the roundoffs were downward, hence the rationale for generally rounding upward. Finally, the slightly unbalanced performance of the rounded set of redundancies can be computed for depth-8 interleaving by again applying the magnification factors to obtain the equivalent ideal interleaving fractional redundancies and then interpolating between ideal interleaving curves at adjacent even-integer redundancies.

Figure 6 shows that for testable SERs between 10^{-2} and 10^{-5}, the depth-8 error magnification factor stays within a small range less than 1.11. The error magnification factor increases with decreasing SER but at a decreasing rate. In all cases plotted, it appears to be leveling off by the time it reaches an SER of 10^{-5}; it is not unreasonable to presume that this leveling off will continue through the untestable values of SER around 10^{-7}. The error magnification factors also increase with increasing codeword redundancy $2E$, but appear to increase very slowly for E above 10. Nominal depth-8 error magnification factors for the target SER around 10^{-7} have been inferred by extrapolating the family of curves in Fig. 6. The

values used for this analysis are 1.13 for $E = 48 \pm 6$, 1.125 for $E = 32 \pm 4$, 1.12 for $E = 16 \pm 2$, 1.105 for $E = 10 \pm 1$, and 1.09 for $E = 5 \pm 1$.

Table 3 shows the results of this refined analysis procedure for four-stage decoding. At various candidate design point values of E_b/N_o for the software Viterbi decoder, a balanced set of fractional redundancies is obtained to yield an overall SER of 2×10^{-7} with ideal interleaving. The nominal error magnification factors for depth-8 interleaving are applied to the redundancies for ideal interleaving, and the resulting depth-8 redundancies are rounded to even integers. The corresponding SER is computed from the curves for infinite interleaving, again using the nominal error magnification factors. The overall signal-to-noise ratio for the concatenated code is then computed by adding the overhead imposed by the selected redundancies.

Table 3. Design values of redundancies for various possible operating points of the S/W Viterbi decoder, with redundancies a, b, c, d repeated according to pattern (a, d, c, d, b, d, c, d).

Design S/W decoder operating point*	Resulting concatenated code E_b/N_o, dB	Balanced redundancies for ideal interleaving				Assumed error magnification factors for depth-8 interleaving				Design redundancies for depth-8 interleaving				Resulting SER
		a	b	c	d	M_a	M_b	M_c	M_d	a	b	c	d	
−0.10	0.58	98.52	61.53	30.67	9.79	1.13	1.125	1.12	1.09	110	70	34	12	1.9×10^{-7}
−0.05	0.58	90.74	57.71	28.91	9.33	1.13	1.125	1.12	1.09	104	66	32	10	1.8×10^{-7}
0.00	0.58	83.14	53.96	27.02	8.88	1.13	1.125	1.12	1.09	94	60	30	10	2.1×10^{-7}
0.05	0.59	76.27	49.89	25.89	8.57	1.13	1.125	1.12	1.09	86	56	28	10	2.3×10^{-7}

* Convolutional code E_b/N_o, dB.

Note that essentially identical concatenated code design points just under 0.60 dB are obtained over a range of convolutional code design points from −0.10 to +0.05 dB, each using a custom-designed set of optimum redundancies. The set of redundancies listed in Table 3 for a convolutional code design point of 0.00 dB is the same as those discussed in the earlier example. There are many sets of "optimum" redundancies that achieve essentially the same performance. Table 4 lists 24 different redundancy sets that all produce an average SER of 2×10^{-7} at a concatenated code signal-to-noise ratio of 0.58 dB. As in [1], the recommendation in this article is to select the optimum redundancy set with the least spread in redundancies and the highest convolutional code operating point. This set is the one listed in Table 3 for a convolutional code design point of 0.00 dB, with redundancy pattern $(a, d, c, d, b, d, c, d) = (94, 10, 30, 10, 60, 10, 30, 10)$.

The foregoing procedure for selecting a set of redundancies has the advantage of allowing a major part of the analysis to take place without any assumptions about how to extrapolate the depth-8 SER performance data to the 10^{-7} range. This makes it possible to isolate and somewhat quantify the inaccuracies that might result from extrapolation. One might design a conservative set of redundancies for depth-8 interleaving by applying an extra-conservative set of magnification factors. This would require an easily calculable increase in the concatenated code signal-to-noise ratio. At concatenated code operating points just under 0.60 dB, an increase of all magnification factors by 0.05 above the nominal magnification factors costs just 0.03 dB in added overhead; an underestimate this large seems unlikely, as it would put three of the magnification factors above the top edge of the graph in Fig. 6. Designing for the adverse magnification factors would correspond to using $a, b, c, d = 98, 64, 32, 10$, instead of the nominal design, $a, b, c, d = 94, 60, 30, 10$, listed in Table 3 for a convolutional code design point of 0.00 dB. Conversely, once a set of depth-8 redundancies has been selected, the sensitivity of the predicted SER to the extrapolation assumptions could be tested by varying the assumed magnification factors for the final performance

Table 4. Various optimal redundancy sets a, b, c, d, repeated according to the pattern (a, d, c, d, b, d, c, d), that achieve SER $= 2 \times 10^{-7}$ at a concatenated code signal-to-noise ratio of 0.58 dB.*

Codeword redundancies				Signal-to-noise ratios, dB		SER
a	b	c	d	Concatenated	Convolutional	
94	60	30	10	0.58	0.00	2.0×10^{-7}
94	62	30	10	0.58	−0.00	2.0×10^{-7}
96	60	30	10	0.58	−0.00	2.0×10^{-7}
96	62	30	10	0.58	−0.00	2.0×10^{-7}
102	64	32	10	0.58	−0.03	2.0×10^{-7}
102	64	34	10	0.58	−0.04	2.0×10^{-7}
102	66	32	10	0.58	−0.04	2.0×10^{-7}
102	66	34	10	0.58	−0.05	2.0×10^{-7}
102	68	32	10	0.58	−0.05	2.0×10^{-7}
102	68	34	10	0.58	−0.05	2.0×10^{-7}
102	70	32	10	0.58	−0.05	2.0×10^{-7}
104	64	32	10	0.58	−0.04	2.0×10^{-7}
104	64	34	10	0.58	−0.05	2.0×10^{-7}
104	66	32	10	0.58	−0.05	2.0×10^{-7}
104	66	34	10	0.58	−0.05	2.0×10^{-7}
104	68	32	10	0.58	−0.05	2.0×10^{-7}
104	70	32	10	0.58	−0.05	2.0×10^{-7}
106	64	32	10	0.58	−0.04	2.0×10^{-7}
106	64	34	10	0.58	−0.05	2.0×10^{-7}
106	66	32	10	0.58	−0.05	2.0×10^{-7}
106	66	34	10	0.58	−0.06	2.0×10^{-7}
106	68	32	10	0.58	−0.05	2.0×10^{-7}
108	64	32	10	0.58	−0.05	2.0×10^{-7}
108	66	32	10	0.58	−0.05	2.0×10^{-7}

* The first listed redundancy set was chosen to support the Galileo LGA mission.

evaluation over a range of reasonable values. For example, it can be shown that the required operating point of the code nominally designed for 0.00 dB would increase to 0.04 dB if all the error magnification factors were increased by 0.05 above the nominal factors. Thus, the design mismatch only costs an additional 0.01 dB above the 0.03 dB that would accrue if the adverse magnification factors could be anticipated. Because of this relative insensitivity of the code's performance to the exact design parameters, the nominal design was recommended and is being implemented for the Galileo LGA mission.

VI. Four-Stage Redecoding Dynamics: An Example

Figure 7 depicts an example of how the four-stage redecoding process works. The block of eight Reed–Solomon codewords, with error correction capabilities (47, 5, 15, 5, 30, 5, 15, 5), is shown in five snapshots. The first snapshot depicts the bursts of errors emanating from the first-stage of Viterbi decoding before any Reed–Solomon decoding. The 8-bit symbol errors output from the Viterbi decoder are represented by the black left-to-right traces. Correctly decoded symbols occupy the gray regions of the code

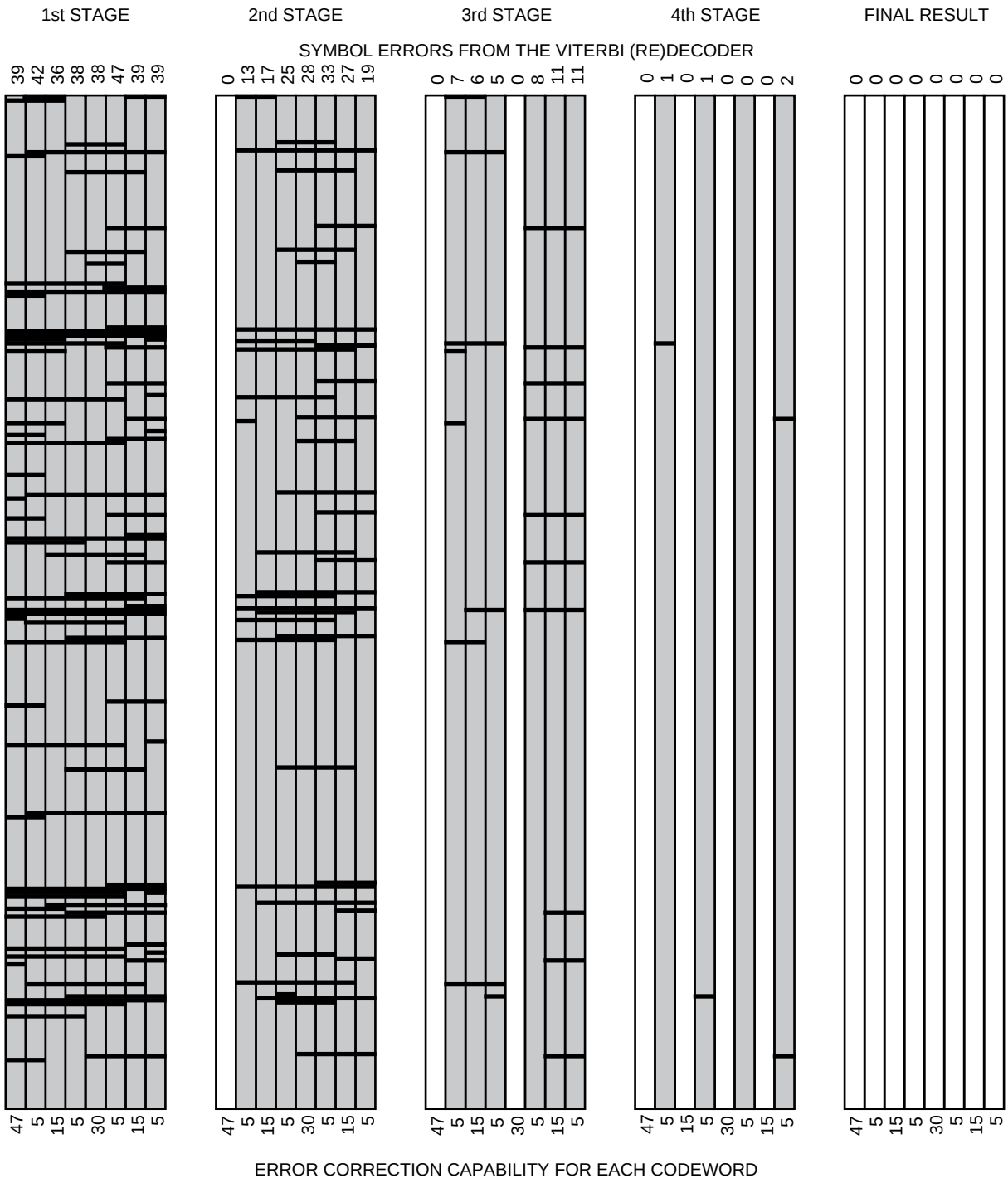

Fig. 7. Illustration of four-stage redecoding dynamics for a sample code block.

block. Shown at the top of the block are the symbol error counts in the individual codewords. These range from 36 to 47, making all the codewords undecodable except for the one with the highest redundancy. The second snapshot shows the code block after the first codeword is corrected by the first-stage of Reed–Solomon decoding. The corrected codeword, depicted in white, now has zero errors and is fed back to assist the second stage of Viterbi decoding. The output of the Viterbi redecoder is improved by the known

symbols from the one known codeword, and the resulting error bursts are thinned out and shortened, as shown in the second snapshot. Now the codeword with correction capability 30 is barely decodable despite 28 errors, so this codeword has zero errors in the third snapshot. With only three unknown symbols between pairs of known symbols, the output from the third-stage Viterbi redecoder is improved to the point where both codewords with correction capability 15 can be decoded by the Reed–Solomon decoder. Finally, in the fourth snapshot, with four known symbols out of every eight, the fourth-stage Viterbi redecoder output sports only occasional isolated symbol errors, which are easily corrected by the final stage of Reed–Solomon decoding despite the low correction capability of the fourth-stage codewords.

This example was obtained from simulated data that were intentionally run at several tenths of a dB below the threshold E_b/N_o required to achieve a BER of 10^{-7}, because, at the design threshold, the Viterbi (re)decoder error bursts would have been sparse enough to make the illustration unenlightening. The choice of a below-threshold operating point also demonstrates another facet of the four-stage decoding process. As seen in the first two snapshots, at this low E_b/N_o, the first two codewords are very lucky to be decodable; in fact, some neighboring codewords have error counts equaling or exceeding the error correction capabilities of the first- and second-stage codewords. This emphasizes that the performance of the high-redundancy codes breaks down very rapidly as E_b/N_o is reduced below the design threshold, whereas the lower redundancy codes used in the third and fourth stages are relatively unaffected by a few tenths of a dB reduction.

VII. A Caveat: Undetected Errors

Throughout this analysis and that of [1], it has been assumed that Reed–Solomon codewords are always either correctable or undecodable. The possibility of undetected Reed–Solomon errors has not been considered. This has traditionally been a safe assumption for codes with large correction capabilities E, because from [3] the undetected error rate is bounded by $(1/E!)$ times the detected error rate. However, for the fourth-stage codewords with $E = 5$, the undetected error rate can be up to 10^{-2} times the detected error rate, and so the possibility of undetected errors cannot be ignored.

Undetected errors in the four weakest codewords do pose a real threat if any attempt is made to decode these words before the reliability of the Viterbi redecoder output is strengthened by having every other symbol known, as shown in the next-to-last snapshot in Fig. 7. Conversely, however, if the weakest codewords are always decoded subsequent to the final stage of Viterbi redecoding based on known symbols from all of the four stronger codewords, both detected and undetected errors are so rare that they do not breach the overall BER requirement of 10^{-7}. If E_b/N_o is reduced to the point where this assumption is no longer valid, the stronger codewords become undecodable first, and the fourth stage of decoding is never reached.

The following caveat suggests a very safe, conservative decoding algorithm that always utilizes exactly four stages as described in this article: “Decode no word before its time.” Such a decoder takes four times as long to decode as a corresponding one-stage decoder. However, this extreme conservatism is unnecessary because the four codewords with correctabilities 47, 15, 30, and 15 do in fact detect their errors almost always. Therefore, it is safe to allow these codewords to be decoded as early as possible, regardless of whether the corresponding Viterbi (re)decoder output has been cleaned up by the successful decoding of stronger codewords in previous stages. The important caveat is that the four weakest codewords should never be decoded until the Viterbi redecoder utilizes information from all four of the stronger codewords. As long as this restriction is honored, there will be essentially no change in the overall output BER. Yet the modified algorithm can allow for a probabilistic speedup in decoding time, sometimes requiring four stages, three stages, or two stages, but never one stage.

VIII. Summary of Performance Results

Table 5 summarizes the performance results discussed above for four-stage decoding and compares them to previous results for one- and two-stage decoding. For a fair comparison, the one- and two-stage

SERs are recalculated here using the new software decoder calibration curves and the same assumed error magnification factors for depth-8 interleaving. The required signal-to-noise ratios for one- and two-stage decoding are lower than the values quoted in [1] by 0.03 and 0.01 dB, respectively.

Table 5. Performance comparisons for depth-8 interleaving at SER $= 2 \times 10^{-7}$, assuming no Reed–Solomon redecoding using erasure declarations.

Decoding stages	1	2	4
Codeword redundancies	(32,32,32,32,32,32,32,32)	(66,20,22,20,66,20,22,20)	(94,10,30,10,60,10,30,10)
Convolutional code E_b/N_o	0.56 dB	0.19 dB	0.00 dB
Concatenated code E_b/N_o	1.14 dB	0.77 dB	0.58 dB

The results in Table 5 do not include any effects from utilizing Reed–Solomon erasure declarations. As noted earlier, the performance improvement at an SER of 2×10^{-7} for a Reed–Solomon decoder that makes use of erasure declarations is roughly 0.19 dB for one-stage decoding, 0.02 dB for two-stage decoding, and 0.00 dB for four-stage decoding.

Two-stage decoding without erasure declarations is worth 0.37 dB relative to a baseline of one-stage decoding without erasure declarations. Adding erasure declarations gains another 0.02 dB for a total improvement of 0.39 dB. Four-stage decoding, with or without erasure declarations, gains 0.56 dB relative to the baseline and 0.17 or 0.19 dB relative to two-stage decoding with or without erasure declarations, respectively.

Uncertainties in the performance estimates stem mostly from the lack of enough data to directly verify decoded SERs around 10^{-7} with depth-8 interleaving. To first order, errors of this type are likely to affect performance predictions for one-, two-, and four-stage decoding in the same direction; hence, comparisons are not likely to change much. In absolute terms, the adverse uncertainty in four-stage decoding performance is likely to be less than 0.04 dB. The favorable uncertainty due to this effect is slightly smaller, as are the adverse uncertainties for one- and two-stage decoding. As mentioned earlier and in [1], there is an additional favorable uncertainty of a few hundredths of a dB for the multiple-stage decoding cases only, due to the technique of substituting "equivalent" BVD data with the same average SER but less benign burst characteristics, in analyzing the second, third, and fourth decoding stages. The magnitude of this effect has not been assessed, but it might provide an argument for adding a couple of extra redundant symbols to the strongest codeword only, in order to maintain a balanced codeword set if the weaker (redecoded) codewords achieve SERs slightly better than predicted.

As noted earlier, if E_b/N_o drops below the threshold designed to produce a BER of 10^{-7}, the performance of the highest redundancy Reed–Solomon codes falls apart, and the decoding of the interleaved code block never gets started. The overall BER increases dramatically according to the steep slope of the high-redundancy code performance curves. Conversely, if E_b/N_o is increased above the design threshold, further reduction in overall BER below 10^{-7} is hampered by the flatter slope of the performance curve for the four weakest codewords. Figure 8 shows the unusual performance curve that characterizes the four-stage Galileo LGA decoding system. Also shown for comparison are performance curves for the two-stage system analyzed in [1] and the standard one-stage concatenated system with a constant redundancy-32 Reed–Solomon code and no Viterbi redecoding. For four-stage decoding, the error rate falls off very steeply as E_b/N_o is increased toward the design threshold; in this region, performance is dominated by that of the highest-redundancy code(s). Upon reaching the design threshold, the performance curve flattens out; here the dominant error contribution comes from the weakest codewords. The lesson learned from considering the entire four-stage performance curve is that you get exactly what you ask for: a very steep descent reaching the required error rate at a minimum expenditure of E_b/N_o, but slow improvement

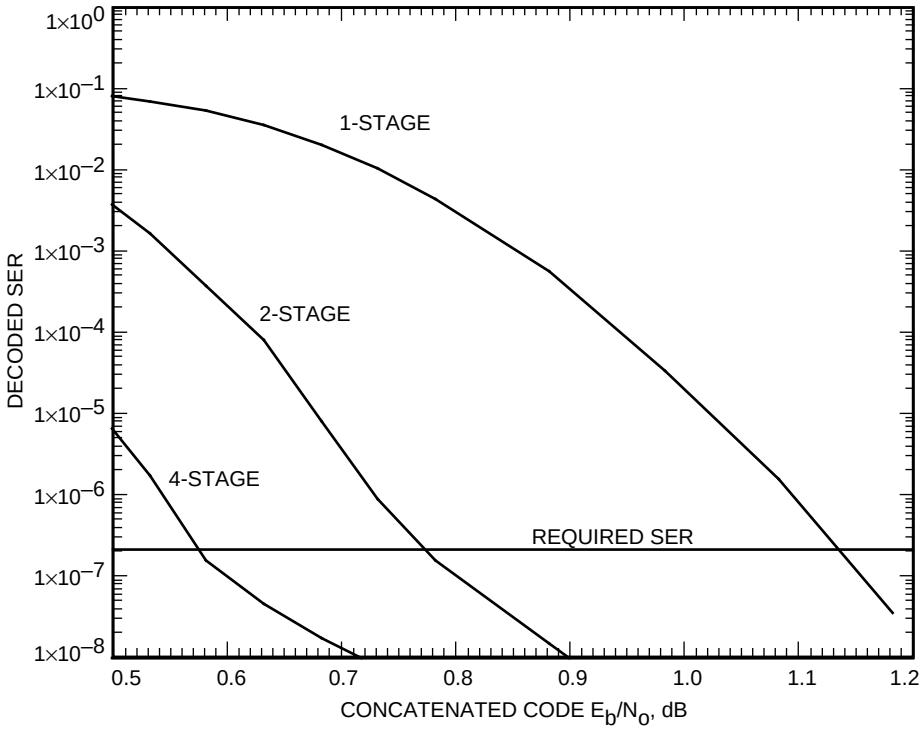

Fig. 8. Performance curves for one-, two-, and four-stage decoding with depth-8 interleaving and near-optimum redundancies.

beyond the requirement if further E_b/N_o is supplied. The same effect is evident but less noticeable for two-stage decoding. For one-stage decoding, the performance curve takes the traditional convex shape. The four-stage performance curve plunges most rapidly to the required SER level, reaching that point 0.56 dB more cheaply than one-stage decoding and 0.17 dB more cheaply than two-stage decoding. On this basis, the Galileo project selected four-stage decoding as the system for maximizing the possible data return.

References

[1] S. Dolinar and M. Belongie, "Enhanced Decoding for the Galileo S-Band Mission," *The Telecommunications and Data Acquisition Progress Report 42-114, April–June 1993*, Jet Propulsion Laboratory, Pasadena, California, pp. 96–111, August 15, 1993.

[2] O. Collins and M. Hizlan, "Determinate-State Convolutional Codes," *The Telecommunications and Data Acquisition Progress Report 42-107, July–September 1991*, Jet Propulsion Laboratory, Pasadena, California, pp. 36–56, November 15, 1991.

[3] R. J. McEliece and L. Swanson, "On the Decoder Error Probability for Reed–Solomon Codes," *The Telecommunications and Data Acquisition Progress Report 42-84, October–December 1985*, Jet Propulsion Laboratory, Pasadena, California, pp. 66–72, February 15, 1986.