Long Decoding Runs for Galileo's Convolutional Codes

C. R. Lahmeyer and K.-M. Cheung
Communication Systems Research Section

Decoding results are described for long decoding runs of Galileo's convolutional codes. A 1-kbit/sec hardware Viterbi decoder is used for the (15, 1/4) convolutional code, and a software Viterbi decoder is used for the (7, 1/2) convolutional code. The output data of these long runs are stored in data files using a novel data compression format which can reduce file size by a factor of 100 to 1 typically. These data files can be used to replicate the long, time-consuming runs exactly and are useful to anyone who wants to analyze the burst statistics of the Viterbi decoders. The 1-kbit/sec hardware Viterbi decoder has been developed in order to demonstrate the correctness of certain algorithmic concepts for decoding Galileo's experimental (15, 1/4) code, and for long-constraint-length codes in general. The hardware decoder can be used both to search for good codes and to measure accurately the performance of known codes.

I. Introduction

Many long decoding runs of 5 to 40 Mbits have been performed for Galileo's experimental (15, 1/4) convolutional code and Galileo's standard (7, 1/2) convolutional code. A 1-kbit/sec hardware Viterbi decoder[1] is used for the (15, 1/4) convolutional code, and a software Viterbi decoder on the RTOP71 Sun-3/260 computer is used for the (7, 1/2) convolutional code. The output data from these long runs are stored in data files using a novel data compression scheme of retaining only the decoded ones and storing them in hexadecimal form. With this format a typical output compression of 100 to 1 is achieved. These data files can be used to replicate the long, time-consuming runs exactly and are useful to anyone who wants to analyze the burst statistics of the Viterbi decoders.

[1]C. R. Lahmeyer, "The 1 Kilobit per Second Viterbi Decoder," Interoffice Memorandum 331-88.3-042, Jet Propulsion Laboratory, Pasadena, California, August 19, 1988.

A 1-kbit/sec hardware Viterbi decoder was developed in the past few months in order to demonstrate the correctness of certain algorithmic concepts in the decoding of long-constraint-length convolutional codes. At present this decoder is designed to decode any convolutional code of constraint length 15 and with code rate $1/n$ as low as 1/6. Most of the recent test runs have used the (15, 1/4) convolutional code that the Galileo project has selected [1]. It has the following generator polynomials:

$$46321 = 100\ 110\ 011\ 010\ 001$$
$$51271 = 101\ 001\ 010\ 111\ 001$$
$$63667 = 110\ 011\ 110\ 110\ 111$$
$$70535 = 111\ 000\ 101\ 011\ 101$$

Since the fast hardware Viterbi decoder is at present configured to decode convolutional codes of constraint length 15 only, a software Viterbi decoder was developed in the RTOP71 Sun computer to perform long decoding runs for

Galileo's standard (7, 1/2) convolutional code. The standard code has the following generator polynomials:

$$133 = 1\ 011\ 011$$

$$171 = 1\ 111\ 001$$

The initial motives for performing the long decoding runs were to facilitate the study of the proper interleaving depth for the Reed-Solomon code used by Galileo, and to develop theoretical models (e.g., the geometric burst model [2]) for the decoded output of the Viterbi decoder, from which concatenated code performance can be accurately estimated without directly simulating the entire concatenated system. These decoding run outputs represent a data base useful to anyone studying the burst nature of the output error patterns of the Viterbi decoders.

II. Description of Test Setup of the Hardware Decoder

While the (7, 1/2) convolutional code is decoded entirely in software, the long decoding runs for the (15, 1/4) convolutional code require the use of the above-mentioned hardware decoder in combination with software. The test configuration used for the decoding runs is shown in Fig. 1. The hardware decoder interfaces with a PC-compatible computer. Software in the PC generates the test data and transmits it to the hardware decoder, where the most computationally intensive part of the decoding, called metric computation, is performed. The PC then performs the part of the decoding process called traceback, in order to complete the decoding. The PC then condenses the decoded bits into the compressed form and archives the results to hard disk.

The generation of the source data is performed by a software noise generation routine in the PC. Gaussian noise symbols are generated in software at a user-selectable noise level. The noise symbols are quantized to 8-bit sign-magnitude representation. The information content of this data is assumed to be all zeros; thus, any nonzero decoded bits represent decoding errors. This is the usual convention when running the decoder to test code performance, and it is theoretically justified because the code is linear and because it has been shown that the decoder does not favor zeros in any way.

III. Data Representation Scheme

A typical method of representing decoded output is to print ASCII 1's and 0's for all the decoded bits, but such an approach would produce a 5-Mbyte DOS file for a 5-Mbit decoding run. Therefore a compact representation scheme was developed which preserves all of the information about the decoder output but reduces file size by a factor of 100 to 1 typically. This scheme relies on the fact that the information content is all 0's, and thus the vast majority of the decoded bits will be 0's, with only a few 1's representing decoding errors. Using a scheme somewhat like spacecraft image compression, only the "changes," or in this case the error bursts, are printed. These are represented in hexadecimal notation.

Figure 2 is a sample printout of a decoding run at a 0.45-dB signal-to-noise ratio (E_b/N_0). The first column is a decimal representation of the number of bits between the start of this burst and the start of the previous one. The first burst started at bit number 0 and the second burst starts 381 bits later. Following this is a hexadecimal representation of the error burst itself. For example, 9100 represents a burst of three error bits given as 1001000100000000 in binary. The definition of the end of a burst is a string of at least 16 bits which are all 0's. Four 0's are printed at the end of each burst to act as a delimiter between bursts and to signify the 16 zero bits. This definition is somewhat loose in that some of the last bits of the printed burst can also be 0's, but no information is lost in any case. All decoding errors will ultimately be listed once each. The line with -1 signifies the end of the decoding run. Thereafter follow some statistics about the entire run. Most are self-explanatory, with Pb representing the bit error rate and Ps signifying the symbol error rate, i.e., the fraction of Reed-Solomon symbols (8 decoded bits) that are corrupted. The size of the entire file "bursts.45" is 46 kbytes.

Table 1 lists all the files accumulated so far for the Galileo code given above. Represented here are several runs of 5 Mbits and a few at 20 Mbits or more. The filename indicates the noise level used in that run, e.g., "bursts.45" signifies a run with $E_b/N_0 = 0.45$ dB.

Table 2 lists files recently generated by software decoding of the NASA standard (7, 1/2) code used by Voyager, Galileo, and other missions. In the filename, "7" signifies the constraint length and the next digits give the noise level. For example, "nburst7.1.4" signifies a run with the (7, 1/2) code at $E_b/N_0 = 1.4$ dB.

IV. Conclusions

A library of decoded output data from long decoding runs of Galileo's convolutional codes has been started. Some early runs in this collection are listed here, and it is anticipated that many more runs with different codes and sample sizes will be performed in the future. Any of the files referenced in Tables 1 and 2 can be made available to interested users on request. Useful applications of this work have already been obtained in the study of how the experimental Galileo convolutional code performs when concatenated with the 8-bit (255, 223) Reed-Solomon code [3].

References

[1] S. Dolinar, "A New Code for Galileo," *TDA Progress Report 42-93*, vol. January–March 1988, Jet Propulsion Laboratory, Pasadena, California, pp. 83–96, May 15, 1988.

[2] R. Miller, L. Deutsch, and S. Butman, *On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes*, JPL Publication 81-9, September 1, 1981.

[3] K. Cheung and S. Dolinar, "Performance of Galileo's Concatenated Codes with Nonideal Interleaving," *TDA Progress Report 42-95*, this issue.

Table 1. Decoding runs to date with the (15, 1/4) convolutional code

Filename	Total bits decoded
bursts.0	5 Mbits
bursts.12	5 Mbits
bursts.22	5 Mbits
bursts.3	600 kbits
bursts.32	5 Mbits
bursts.42	5 Mbits
bursts.45	5 Mbits
bursts.5	22 Mbits
bursts.6	20 Mbits
bursts.7	40 Mbits

Table 2. Software decoding runs for the (7, 1/2) convolutional code

Filename	Total bits decoded
nburst7.1.4	10 Mbits
nburst7.1.45	5 Mbits
nburst7.1.5	5 Mbits
nburst7.1.55	10 Mbits
nburst7.1.6	5 Mbits
nburst7.1.65	10 Mbits
nburst7.1.7	5 Mbits
nburst7.1.75	10 Mbits
nburst7.1.8	5 Mbits
nburst7.1.85	10 Mbits
nburst7.1.9	40 Mbits
nburst7.2.0	40 Mbits

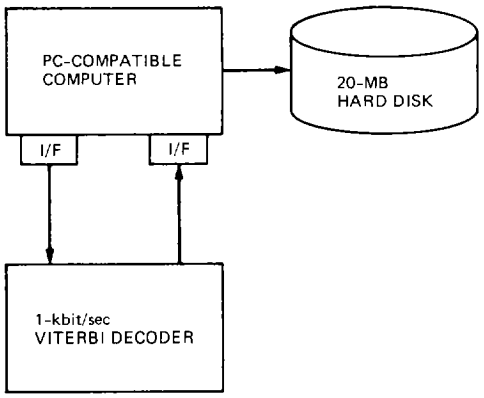

Fig. 1. Configuration for decoding with 1-kbit/sec Viterbi decoder.

```
    0 9100 0000
  381 81aa 5c0b 3a46 8000 0000
11796 bd3f 609a 1d00 0000
 4643 ef3b fd68 0000
 1490 f371 d000 0000
 4531 8afa 5eef 03ec 8000 0000
 4946 aef8 3b8a 4400 0000
 1898 819f e4fa 8c13 5d4c eb20 0000
 9206 8b62 6221 18f9 f400 0000
  612 d38e 8000 0000
 2621 f4ed e3a0 0000
 1556 f5c6 1c00 0000
                    .
                    .
                    .
                    .
                    .
                    .
 5176 c10b dcff 1f14 f258 0ab9 557b 0341 0000
 1467 b630 0000
 6557 f99e 3a00 0000
 6786 81b7 843f 086b e3a0 0000
  185 c100 d9b0 4618 0000
 4962 8a00 0000
  172 c000 0000
 2469 adb2 b389 1d64 e000 0000
 1395 850d 5e96 e755 125b 04fd cec4 9800 0000
 3468 e800 0000
 3715 c727 2720 0000
 1379 8800 0000
  651 9302 a64f 6c00 0000
  405 cc64 c1cc 4b21 f515 b200 0000
   -1

   5000040 Total bits decoded
      1560 Total bursts detected

         bits =  5000040   biterrs =   30964    Pb = 3.425e-003
         syms =   625005   symerrs =    8732    Ps = 8.343e-003
         saturation values = 0

Data recorded at  Eb/No =   0.45  db
```

Fig. 2. Sample decoding run output (from file "bursts.45").