A Software Simulation Study of a Sequential Decoder Using the Fano Algorithm

F. Pollara
Communications Systems Research Section

A set of subroutines has been developed to simulate the performance of a sequential decoder based on the Fano algorithm. This simulation can be used to verify the coding performance of the ICE communication link. The probability of frame deletion can be measured as a function of the number of computations allowed per frame and of E_b/N_o. Both hard and soft quantized inputs are considered.

I. Introduction

While many deep space missions use short constraint length convolutional codes, which can be decoded by the Viterbi algorithm, some missions use long constraint length codes. These codes are efficiently decoded by sequential methods.

ICE (International Cometary Explorer) is an example of a mission using a long constraint length (K = 24) code. The work described in this article was motivated by the need to verify the coding performance of the ICE communication link. The specific convolutional encoder (K = 24, rate = 1/2) used in this mission is shown in Fig. 1. The frame length can be varied, but is typically set at 1,024 bits, including a fixed tail pattern of 24 bits. The sequential decoder is based on the Fano Algorithm (Ref. 1), which is reviewed in Appendix A.

II. The Simulation

In order to carry out this simulation, a set of C-language subroutines was written, representing the operation of a coded communication link as shown in the block diagram of Fig. 2, where each block corresponds to a subroutine. A program called "universe" controls the execution of each subroutine and defines the topology of their interconnections. The subroutine "generator" produces sequences of binary random data in blocks of 1,000 bits, plus a fixed tail sequence of 24 bits, which is appended at the end of each block. The subroutine "coder" implements a convolutional encoder as shown in Fig. 1. Then Gaussian noise generated by "gauss" is added to the link by the subroutine "add" in order to simulate a given E_b/N_o. The sequential decoding takes place in the subroutine "seq," which also provides quick-look decoded data; this data is useful in replacing

frames that would otherwise be deleted. The heart of the decoder is a shift register which also contains a replica of the encoder. High speed is achieved by using a pointer to the contents of this register to avoid time consuming read/write operations. Finally, the error statistics are displayed by the subroutine "error."

III. Performance Results and Discussion

Two versions of the sequential decoder subroutine "seq" have been developed, for hard and soft quantized (3-bit) inputs. Each performance measure depends on the following parameters: E_b/N_o, frame and tail length, the increment Δ used to update the running threshold in the Fano algorithm, the maximum number of computations C allowed for each frame, and, for soft decoding, the particular metric used. The number of computations is defined as the total number of forward moves per frame. A frame is deleted when it cannot be decoded in C or fewer computations.

Figure 3 shows the performance of the hard-quantized decoder in terms of probability of frame deletion P_{FD} versus E_b/N_o, for some values of C and metric ratio MR, which is the ratio of metric increments assigned to symbol agreements and disagreements (Ref. 2). Figure 4 shows the performance for the soft-quantized decoder.

The performance results obtained by this software simulation can be compared to those of a hardware decoder only if the speed advantage is the same in both cases. Specifically, the number of forward moves is not equivalent to the number of computations per second in the hardware decoder. Furthermore, the presence of a buffer for frames in the hardware version tends to improve the performance at higher E_b/N_o, as shown in Fig. 4, where $\overline{C}$ is the maximum number of computations allowed to decode the 3 frames in the buffer.

Figure 5 shows P_{FD} vs the maximum number of computations per frame C, for hard and soft-quantized inputs, respectively.

Acknowledgment

The author wishes to thank George Pitt for the generation of numerical results.

References

1. Blahut, R. E., *Theory and Practice of Error Control Codes*, Addison-Wesley, 1983.

2. Clark, G. C. and Cain, J. B., *Error-Correcting Coding for Digital Communications*, Plenum, 1982.

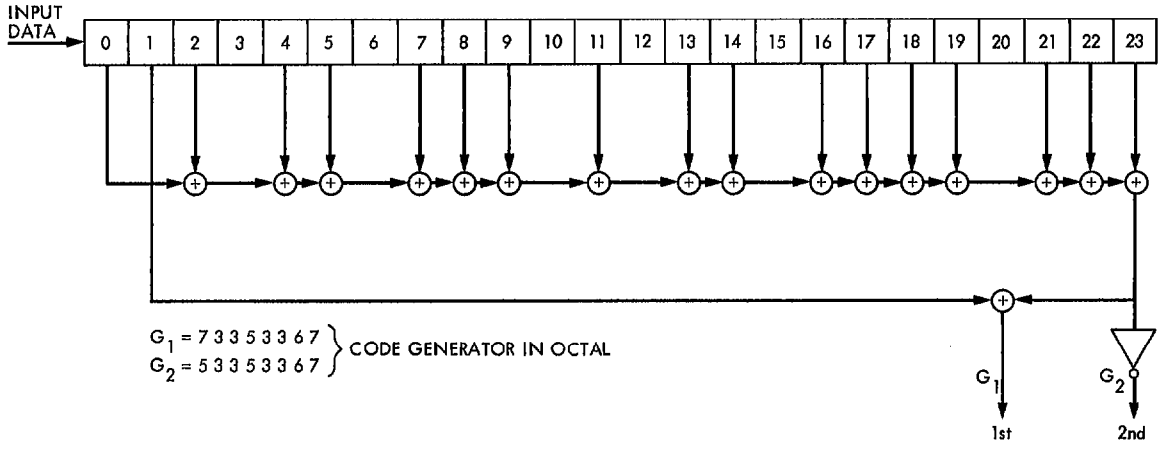

Fig. 1. ICE convolutional encoder (K = 24)

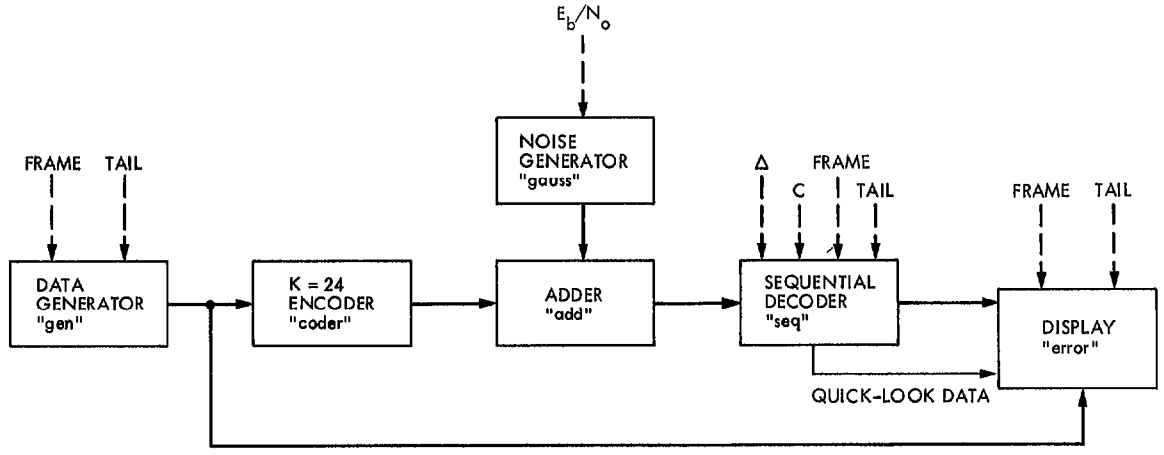

Fig. 2. Sequential decoder simulation block diagram

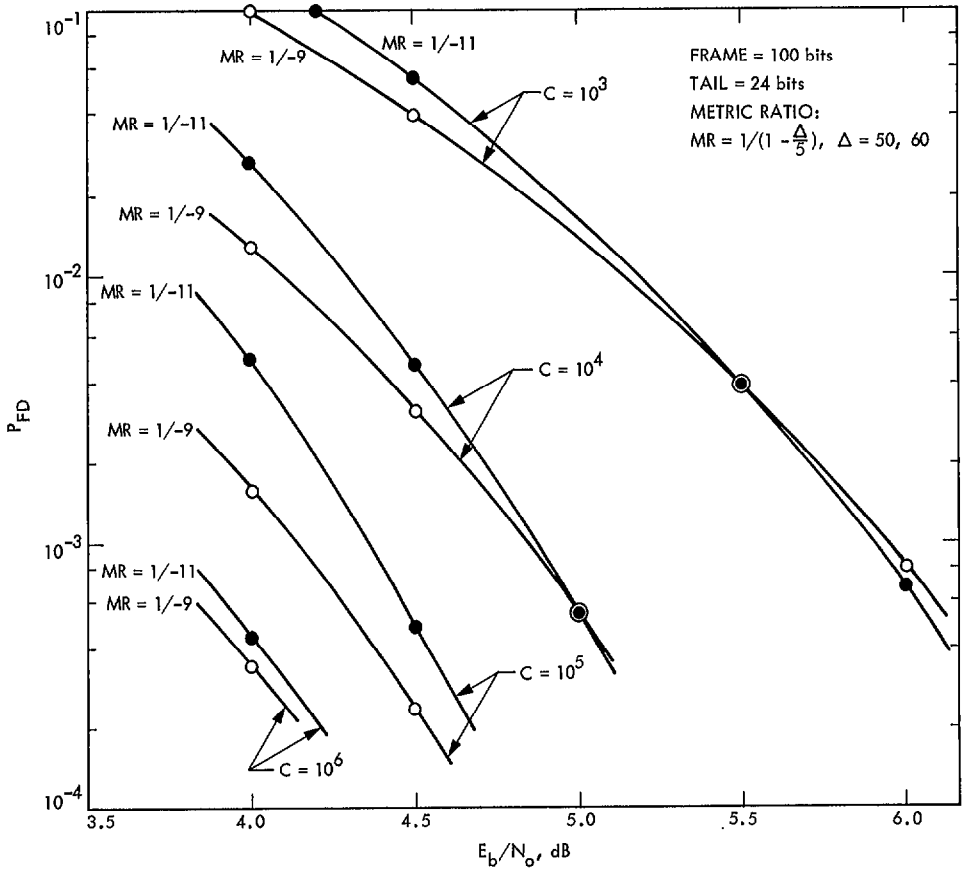

Fig. 3. P_{FD} vs E_b/N_o for hard-quantized decoder

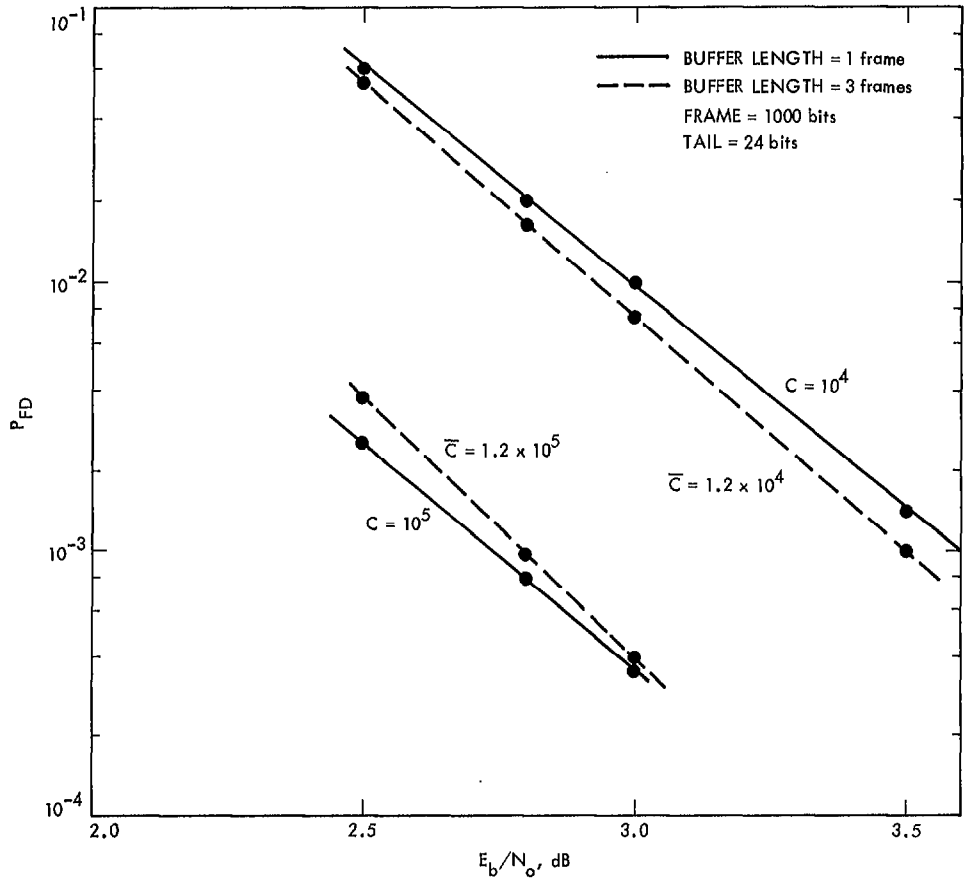

Fig. 4. P_{FD} vs E_b/N_o for soft-quantized decoder ($\Delta = 256$)

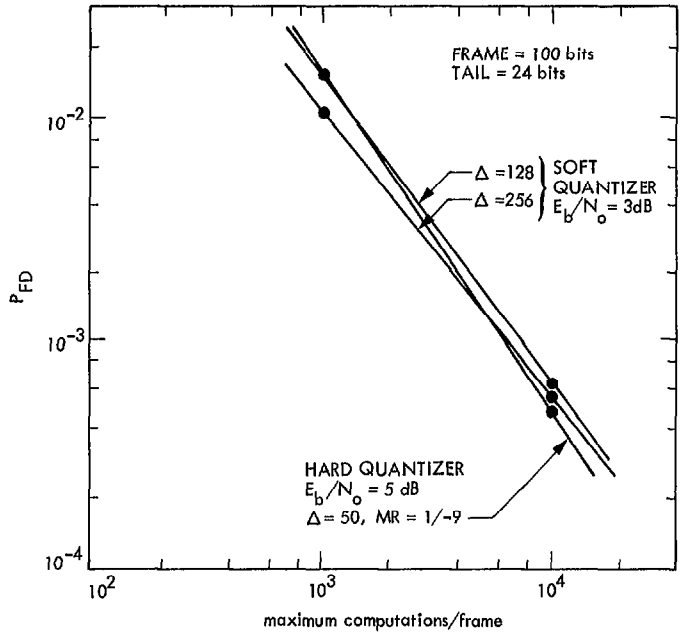

Fig. 5. P_{FD} vs maximum computations per frame

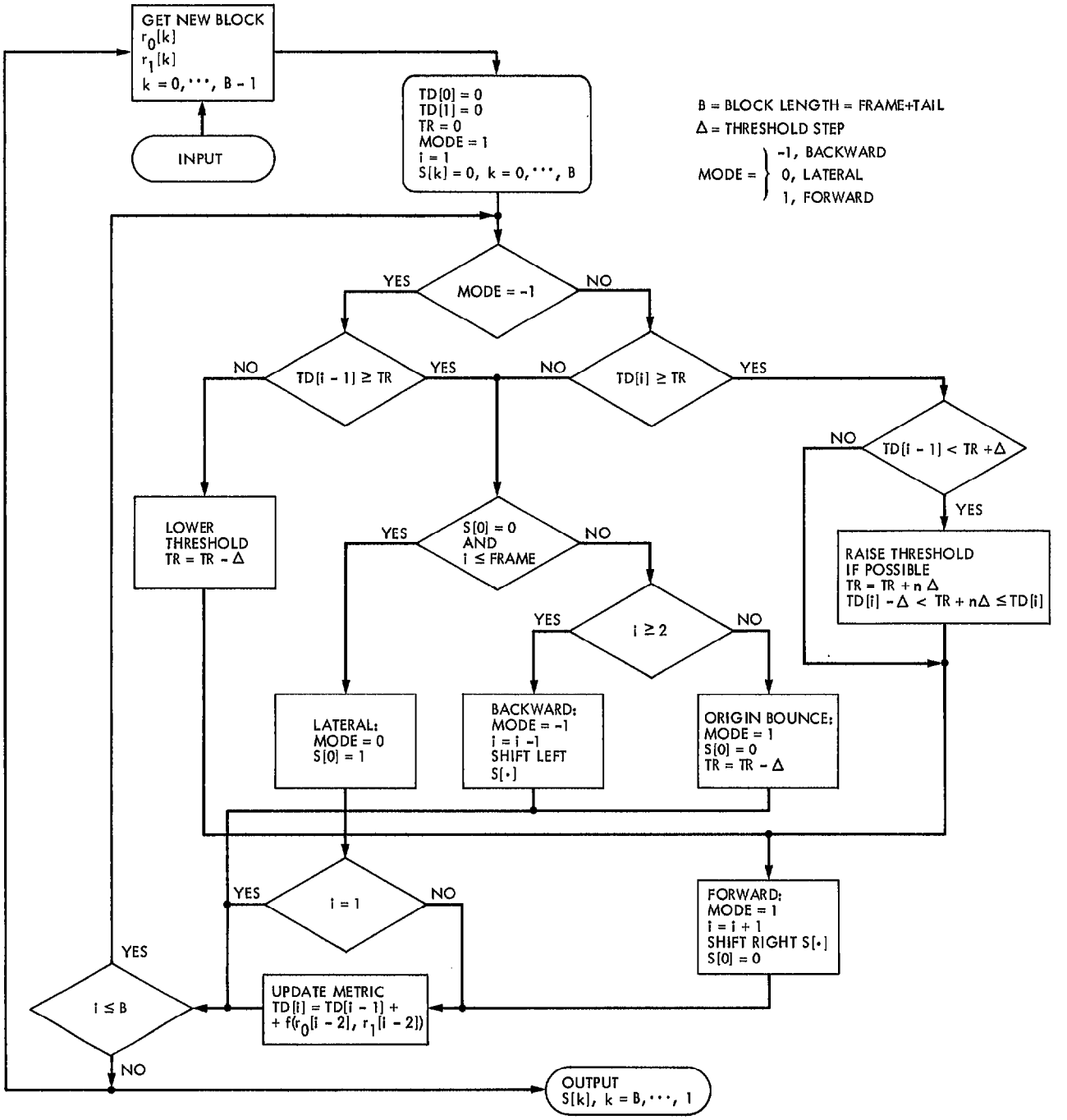

Fig. 6. Sequential decoder flow diagram (hard quantizer)

Appendix A

A brief summary of the Fano algorithm in the form used in this simulation is given for completeness.

Consider the decoding of a frame consisting of FRAME information bits plus TAIL bits in the tail. The task of the decoder is to find the path on a binary tree of length FRAME+TAIL which is the closest approximation to the transmitted frame, according to a given measure criterion (metric). At each node on the tree, the decoder looks forward and attempts a move along the branch corresponding to a "0" information bit. If a "1" was transmitted at that level, the decoder will realize that it is on a wrong path and make a lateral move, which is equivalent to changing the previous attempt into a forward move along the "1" branch. This is the typical behavior of the decoder in the absence of noise.

If there are errors in the received sequence, the decoder will perceive that it is following a wrong path, according to the rules for search described below. Then it will back up a few branches and explore alternative paths, until it finds a satisfactory path of length FRAME+TAIL. At this point the first FRAME bits in the path are taken as the decoded frame sequence. They may still contain errors, even though this kind of error (decoded bit errors) is extremely rare.

If, for some frame, the decoder is forced to move erratically back and forth on the tree up to a certain imposed limit on the number of forward moves, that frame will not be decoded (deleted).

Consider the flow diagram in Fig. 6. The "tilted distance" TD[·] is a function of the distance between the received symbol and the current path through the tree. At high E_b/N_o, TD[·] is an increasing function, and the decoder moves forward (or laterally) on the tree. The decoder decides that TD[·] is increasing or decreasing by comparing it with a "running threshold" TR which is kept as large as possible, but smaller than TD[·], and changes in steps of size Δ. If TD[·] starts to decrease, the decoder will back up and then explore other paths, or even come back to the same node, but with more confidence to get past it (i.e., with a lower TR). Therefore, the decoder will never reach the same node twice with the same threshold, thus preventing infinite loops.