X-750-71-139
PREPRINT

NEW DEVELOPMENTS IN CONVOLUTIONAL ENCODING AND DECODING

THOMAS J. LYNCH

APRIL 1971

GSFC
GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-750-71-139

NEW DEVELOPMENTS IN CONVOLUTIONAL ENCODING AND DECODING

Thomas J. Lynch

April 1971

Presented at NTG/IEEE Coding Conference
Braunschweig, Germany, Oct. 7-9, 1970

NASA Goddard Space Flight Center
Greenbelt, Maryland

NEW DEVELOPMENTS IN CONVOLUTIONAL ENCODING AND DECODING

Abstract

The historical development of convolutional encoding and decoding is outlined and a brief tutorial explanation of the convolutional coding concept is given. The recent progress in encoding and decoding is described by citing specific examples. In particular, in the decoding area, the application of such techniques as: sequential decoding, threshold decoding, and maximum likelihood decoding is described. Some examples of actual communication systems that incorporate convolutional codes are given.

CONTENTS

NEW DEVELOPMENTS IN CONVOLUTIONAL ENCODING AND DECODING

1. INTRODUCTION

This paper will outline the new developments in convolutional encoding and decoding. These developments will be introduced by a brief historical and tutorial description of convolutional codes. Finally, some applications of convolutional codes in present and future systems will be described.

2. BRIEF TUTORIAL DESCRIPTION OF CONVOLUTIONAL CODES

In a convolutional code the check (or parity) bits are computed as a function of information bits as the latter are fed into the encoder serially. The encoder does not recognize information words, as in the case of block codes, but only frames, which are serial groups of information words.

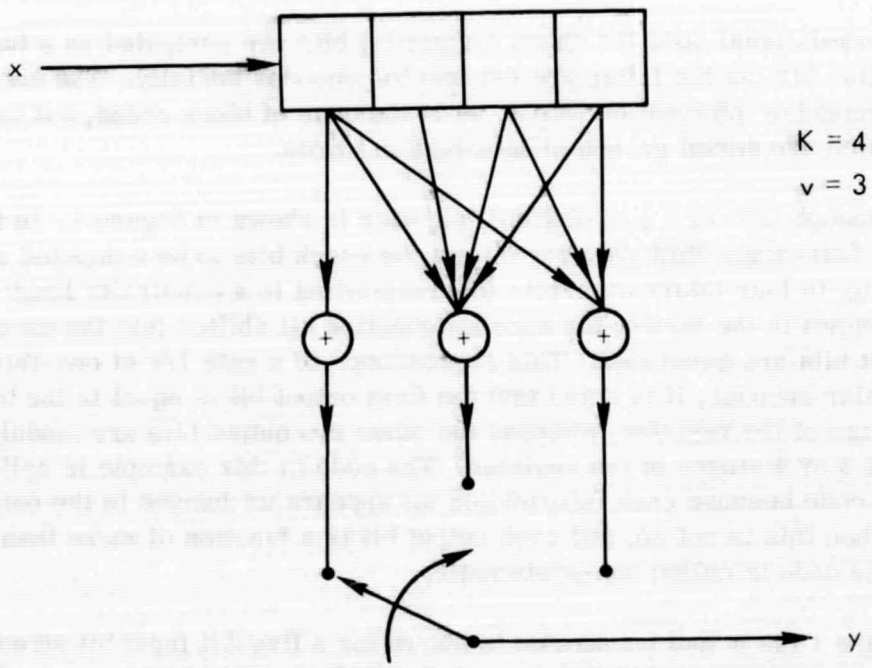
The concept [25] of a convolutional encoder is shown in Figure 1. In this example, a four-stage shift register allows the check bits to be computed as a function of up to four information bits (corresponding to a constraint length, K, of 4). As shown in the figure, for each information bit shifted into the encoder, three output bits are generated. This corresponds to a rate 1/v of one-third. In this particular encoder, it is noted that the first output bit is equal to the bit in the first stage of the register, whereas the other two output bits are modulo-2 functions of 3 or 4 stages of the register. The code in this example is called a systematic code because each information bit appears unchanged in the output bit stream. When this is not so, and each output bit is a function of more than one input bit, the code is called non-systematic.

In Figure 1 the output bit stream is shown for a five-bit input bit stream or frame and corresponds to 9 shifts through the shift register. That is, the first group of 3 output bits corresponds to the shift of the first input bit into the first stage, and the last group of 3 output bits corresponds to the shift of the last information bit out of the last stage. In this example zeros are shifted into the shift register after the last input bit.

For a given encoder arrangement of the type shown in Figure 1, and a given input frame length, a code tree can be constructed. The code tree for the encoder shown in Figure 1 is shown in Figure 2. This code tree is a fundamental concept in convolutional codes, particularly in some of the decoding techniques. The tree shows the encoder output bits for each input, one or zero, at each tree node.

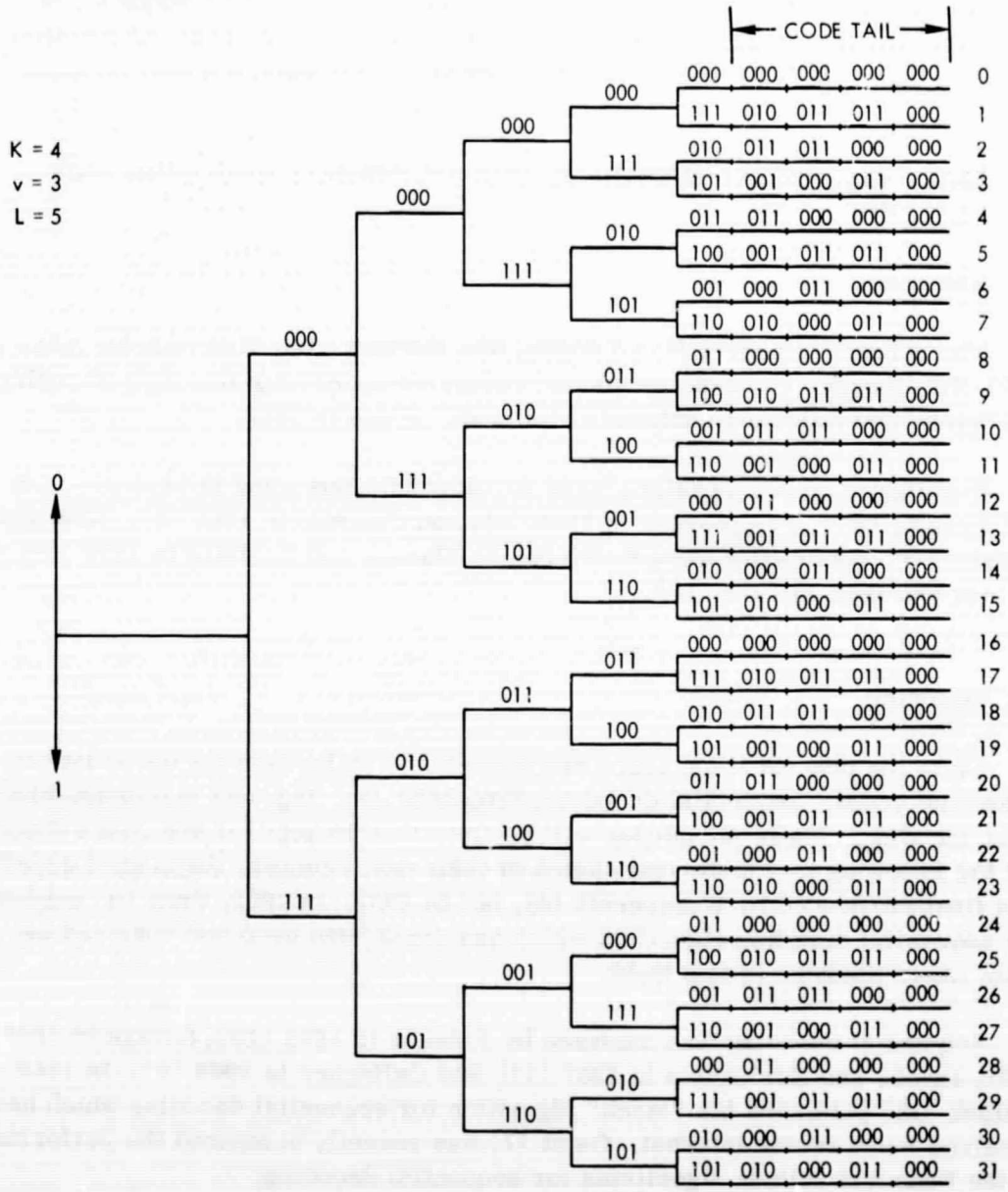
WHEN: x = (1,0,1,1,0) (FOR L = 5)

y = (111, 010, 100, 110, 001, 000, 011, 000, 000)



* From Wozencraft, J. M., and Jacobs, I. M.
"Prinicples of Communication Engineering"
John Wiley & Sons, Inc., N. Y., 1965, pp 410, 411

Figure 1. A Convolutional Encoder.



* From: Wozencraft, J. M. and Jacobs, I. M., "Principles of Communication Engineering", J. Wiley, N. Y., 1965 p g.414.

Figure 2. Convolution Code Tree for the encoder in Figure 1.

The code "tail" is the output bit sequence that results after the last information bit in the frame is shifted into the second, third, etc. stage of the shift register and finally out of the k-th stage. It is customary to load a sequence of zeros into the first stage of the shift register while the end of the information frame is being shifted through, although any known bit sequence may be used.

3. BRIEF HISTORICAL SKETCH OF CONVOLUTIONAL ENCODING AND DECODING

3.1 Encoding

The concept of convolutional coding was introduced by Elias [3] in 1955. In 1959, Hagelbarger [9] used the name "recurrent codes" for this type of coding, and Wyner and Ash [26] analyzed "recurrent" codes in 1963.

Systematic code generators were developed by Bussgang in 1965 [1], Lin and Lyne in 1967 [14], Forney in 1968 [5], and Costello in 1969 [2]. Non-systematic code generators were developed by Massey and Costello in 1970 [18] and Jelinek and Bahl in 1970 [13].

3.2 Decoding

Up to the present time, there are three basic techniques for decoding convolutional codes: sequential decoding, threshold decoding, and maximum-likelihood decoding. These techniques will be described in detail in the next section, but the following is a historical sketch of their development. Sequential decoding was first introduced by Wozencraft [23, 24] in 1957. In 1963, Fano [4] published his sequential decoding algorithm which has since been used and analyzed by many other workers in this field.

Sequential decoding was analyzed by Pinsker in 1965 [19], Savage in 1966 [20], Jacobs and Berlekamp in 1967 [11] and Gallagher in 1968 [6]. In 1969 Jelinek [12] published his "stack" algorithm for sequential decoding which has received much recent interest. Geist [7] has recently compared the performance of the Fano and Jelinek algorithms for sequential decoding.

Threshold decoding for convolutional codes was proposed by Massey [16] in 1963.

A maximum likelihood decoding technique for convolutional codes was proposed by Viterbi [22] in 1967.

4. SOME NEW DEVELOPMENTS

In the development of any new system concept, one would like to find an optimum system and then attempt to build systems that would approach this optimum system. This has not been the case with convolutional encoding and decoding. A convolutional code does not have an algebraic structure, and this has necessitated random coding arguments for theoretical studies (of bounds on error probability, for example) and computer simulations for development of encoding and decoding systems. This explains the name "probabilistic code" used in reference to a convolutional code as compared to the name "algebraic code" used in reference to a block code.

4.1 Encoding

Convolutional encoders have been divided into two types: those that generate systematic codes, and those that generate non-systematic codes. It is common practice to represent the inputs to the modulo-two adders (such as shown in Figure 1) in terms of generating function coefficients.

For every convolutional encoder, we can write a set of generating function coefficients as follows:

$$\begin{aligned} G_1 &= g_{11}\ g_{12}\ g_{13}\ \cdots\cdots\cdots\ g_{1k} \\ G_2 &= g_{21}\ g_{22}\ g_{23}\ \cdots\cdots\cdots\ g_{2k} \\ G_r &= g_{r1}\ g_{r2}\ g_{r3}\ \cdots\cdots\cdots\ g_{rk} \end{aligned} \tag{1}$$

For example, the convolutional encoder in Figure 1 has the following three generating function coefficients:

$$G_1 = 1000$$

$$G_2 = 1111$$

$$G_3 = 1011$$

A systematic code always has one generator function with the following coefficient properties:

$$\text{For } G_1 = g_{11}\ g_{12}\ g_{13} \cdots\cdots\cdots\ g_{1k}$$

$$g_{11} = 1 \tag{2}$$

$$g_{12} = g_{13} = \cdots\cdots\cdots = g_{1k} = 0$$

As noted above, Bussgang [1], Lin and Lyne [14] and Forney [5] developed techniques for finding generator functions for systematic codes with good error-correction properties by maximizing the minimum distance between paths in the code tree. These properties were examined by means of computer simulations, and the resulting generator functions are identical for many constraint lengths. For this reason this type of systematic convolutional code has been given the name: Bussgang-Lin-Lyne-Forney code.

Recently, Costello [2] has developed algorithms for generator functions for convolutional codes with constraint lengths longer than previously obtainable.

In the area of non-systematic codes, there has been considerable interest in developing generator functions since non-systematic codes typically show better performance than systematic codes for the same constraint length K. One of the undesirable characteristics of a non-systematic code, however, is the fact that the input bit sequence no longer appears in the output bit stream, making a "quick-look" capability impossible before decoding.

Massey and Costello [18] developed a non-systematic code which has good performance, and allows "quick look." If D represents the unit delay in one stage of the encoding shift register, then the generating function may be written:

$$\begin{aligned} G_1(D) &= g_{11} + g_{12} D + g_{13} D^2 + \cdots\cdots + g_{1k} D^{k-1} \\ G_r(D) &= g_{r1} + g_{r2} D + g_{r3} D^2 + \cdots\cdots + g_{rk} D^{k-1} \end{aligned} \tag{3}$$

Massey and Costello concentrated on a rate 1/2 non-systematic code with

$$G_1(D) = D + G_2(D) \tag{4}$$

In this way, if the input bit sequence is represented by:

$$I(D) = i_1 + i_2 D + i_3 D^2 + \cdots\cdots\cdots \tag{5}$$

and the output sequences from the two modulo-two adders are:

$$T_1(D) = G_1(D)\ I(D)$$
$$T_2(D) = G_2(D)\ I(D) \qquad (6)$$

then:

$$T_1(D) + T_2(D) = D\ I(D) \qquad (7)$$

Thus by adding the two-bit sequences from the two modulo-two adders, one obtains the original input bit stream shifted by one unit time delay.

As can be seen from Equation 4, the two generator functions have identical coefficients except for the second coefficient.

A type of non-systematic code that has shown notably good performance recently is the "complement" code of Jelinek and Bahl [13]. This code has the following generator properties:

1. It is rate 1/2.
2. The first and last coefficients of both generators are equal to one, or $g_{11} = g_{1k} = g_{21} = g_{2k} = 1$.
3. The other coefficients of G_1 are the complements of the corresponding coefficients of G_2.

4.2 Decoding

4.2.1 Sequential Decoding

4.2.1.1. Fano Algorithm. As mentioned above, the technique of sequential decoding has been dominated by the Fano algorithm (4) since its publication in 1963. The following is a brief description of the Fano algorithm. Basically, this algorithm searches through the code tree (such as the one shown in Figure 2) for the path which has the best agreement in branch symbols with the received sequence of symbols. This "best" agreement is determined at each branch node by means of a function called a "metric" which is based on the logarithm of the likelihood function or L. A. P. which stands for log-a-posteriori-probability. For each symbol transmitted, y, and each symbol received, z, the symbol likelihood function is:

$$L_s = \frac{P(z/y)}{\sum_y P(y)\,P(z/y)} \qquad (8)$$

By quantizing the received symbols (at the output of a demodulator, for example) an improvement in performance of the algorithm is obtained (representing an equivalent coding gain of approximately 2 db). For quantization, Jacobs [10] found that the difference in performance between 8 levels and an infinite number of levels is 0.2 db. The metric that is compared at each branch node is called the branch metric and is equal to the sum of the symbol metrics (which are the same as the log L_s functions except for a bias value). The branch with the higher branch metric is chosen and this metric is added to the sum of previous branch metrics (called the cumulative branch metric) along the path being traced in the code tree. A check is made to see if this cumulative branch metric is greater than a threshold value. This threshold value is increased at each successful node test to bring it closer to (but always less than) the cumulative branch metric. When an incorrect branch has been chosen, the threshold test typically will fail, and the algorithm will successively start back at previous nodes, lowering the threshold if necessary until the threshold test is satisfied and a new forward path is found.

The above forward and backward behavior of the Fano algorithm leads to one of the basic problems of sequential decoding, namely the variable computation time for decoding. The computation time for each frame (corresponding to a complete path through the tree) is a random variable. Various aspects of this random variable have been studied [11, 20], particularly the influence of the channel signal-to-noise ratio. In general, the cumulative probability distribution of the computation time is an exponential Pareto-type function, of the form,

$$P_r (T_c > t_c) = K_c \, t_c^{-\rho} \tag{9}$$

where

K_c is a constant

ρ is the Pareto exponent

The Pareto exponent is defined here in terms of the code rate 1/v and the zero-rate exponent for error probability given by Gallager [6].

When $\rho = 1$, the rate is called the "computation rate," or R_{comp}. As the frame size L goes to infinity, the computation time per decoded digit will remain finite for $R < R_{comp}$, and will tend to infinity for $R > R_{comp}$ [4]. This variable computation time leads to situations in which the sequential decoder cannot keep up with the incoming data rate. In such cases, the sequential decoder has to delete or reject data and resume decoding with the new incoming data. Data deletion, with a corresponding deletion rate is an inherent characteristic of sequential decoding.

Since the introduction of the Fano algorithm, it has been modified slightly by analysts and users, particularly in the use of the threshold test [8], but it still represents a state-of-the-art technique in sequential decoding [21].

4.2.1.2. The Jelinek algorithm [12] is another form of sequential decoding, and the following is a brief description of its operation. A cumulative function called a node value very similar to the cumulative branch metric of the Fano algorithm (since it also includes the likelihood function) is computed for each branch and ordered or "stacked" according to the following procedure:

1. Set origin node value to zero and place it in the stack.
2. Compute the node values of the two successor nodes of the node whose node value is at the top of the stack, and add them to the stack in an ordered manner.
3. Delete from the stack the node value of the node whose successor nodes were just used.
4. If the new top node is in the last level of the tree, stop, and select the path determined by this last node. If not, go back to step 2.

It can be seen that the Jelinek algorithm, in effect, can look back more than one node at a time, in the case of noisy data; whereas the Fano algorithm looks back one node at a time in the same situation. However, at each node, the computation required by the Jelinek algorithm is more complex than the Fano since it involves a search of the entire node value stack. These tradeoffs are involved in the comparison of these two algorithms from the standpoint of computation time, and Geist [7] has made such a comparison - empirically. His results show almost equal computational complexity between the two algorithms, with the Jelinek decoder slightly faster on noisier channels.

4.2.2. Threshold decoding [16] has been called by its author, Massey, "a simple, but in general, non-optimum solution to the decoding problem based on a special subset of parity checks." [17] There are two types of threshold decoding: majority decoding and APP (A Posteriori Probability) decoding. In each case, a set (A_i) of J parity checks is used which is orthogonal on the noise bit e_0 corresponding to the information bit being decoded. A parity check set is called orthogonal on a noise bit e_0 if e_0 is checked by each member of the set, but no other noise bit is checked by more than one member of the set. The general rule for threshold decoding is: choose $e_0 = 1$ if, and only if,

$$\sum_{i=1}^{J} w_i A_i > T \qquad (10)$$

where

w_i are weighting factors, T is the threshold

For majority decoding,

$$w_i = 1, \quad T = \frac{J}{2}$$

for APP decoding

$$w_i = 2 \log \frac{q_i}{p_i}, \qquad T = \frac{1}{2} \sum_{i=0}^{J} w_i$$

where $p_i = 1 - q_i$ is the probability of an odd number of "ones" in the noise bits exclusive of e_0 which are checked by A_i. Compared to sequential decoding, threshold decoding is basically simpler to implement, but requires codes that have the parity check orthogonality described above. The computation time is a constant in the case of threshold decoding as compared to the variable computation time of sequential decoding, and there is no data deletion.

4.2.3. The Maximum Likelihood approach to decoding of convolutional codes is represented by the Viterbi algorithm [22]. Theoretically, the maximum likelihood decoder would compute and compare the likelihood functions for all paths through the code tree, given a received frame, and choose as the correct path the one with the largest likelihood function.

This would involve prohibitive computation time. The Viterbi algorithm is a modified version of this approach with a greatly reduced computation time.

The Viterbi algorithm utilizes the repetitive nature of a code tree: a code tree repeats its structure after K nodes, where K is the constraint length. In operation, this algorithm computes the likelihood function of the 2^K paths from the origin to the Kth node in the code tree. The repetitive structure of the code tree allows the elimination of 2^{K-1} paths at the Kth node level since beyond the Kth node level, the code symbols on branches emanating from certain pairs of branches are identical. The 2^{K-1} paths selected on the basis of maximum likelihood are called the "survivors." In the next step the likelihood functions of the 2^K paths from the origin to the $(K + 1)^{th}$ node are computed and 2^{K-1} paths are eliminated as above. In each step, the algorithm computes 2^K likelihood functions since 2^{K-1} branches are added at the $(K + 1)^{th}$ node to the 2^{K-1} survivor paths. The decoder considers 2^{K-1} candidate paths as it moves through the tree, and finally selects one path at the end of the tree. This selection is really

a "narrowing-down" process which starts at the Lth node((see Figure 2). At this tree level, zeros as fed into the encoder shift register and branching ceases - the code "tail" is generated. With each zero encoder input, a decision is made in the decoder algorithm, reducing the number of candidate paths by a factor of two. Since the code tail is K levels long, then by the time the decoder reaches the last level, it has reduced the 2^{K-1} candidate paths by a factor of 2^{K-1}, or in other words, selected one path through the tree.

5. SOME APPLICATIONS OF CONVOLUTIONAL CODES

Because of their high coding gain, convolutional codes have been proposed for use in deep space communication [10].

A convolutional code was used on the PIONEER IX spacecraft launched on November 8, 1968 [15] into a heliocentric orbit. The code was a systematic, rate 1/2 code with a constraint length of 25 bits and a frame length of 224 bits. The decoder was a Fano algorithm type programmed for a medium size computer (SDS 920).

Four future NASA spacecraft will use convolutional codes: IMP (Interplanetary Monitoring Platform) H, I and J; and RAE (Radioastronomy Explorer) B. The RAE-B spacecraft will be in lunar orbit.

It is planned to use a convolutional code on the German-U.S.-HELIOS spacecraft which will be launched in 1974.

ACKNOWLEDGMENT

The author wishes to acknowledge the helpful suggestions of Professor James L. Massey in the preparation of the material for this paper.

REFERENCES

1. Bussgang, J. J.: Some properties of binary convolutional code generators. IEEE Transactions on Information Theory, IT-11, January 1965, pp. 90-100.

2. Costello, D. J.: A construction technique for random error-correcting convolutional codes. IEEE Transactions on Information Theory, IT-15, September 1969, pp. 631-636.

3. Elias, P.: Coding for noisy channels. IRE Convention Record-Part IV, 1955, pp. 37-44.

4. Fano, R. M.: A heuristic discussion of probabilistic decoding. IEEE Transactions on Information Theory, IT-9, April 1963, pp. 64-74.

5. Forney, G. D.: Final report on a study of a simple sequential decoder. U. S. Army Satellite Communication Agency Contract DAA-B07-68-C-0093, Appendix A. Codex Corporation, Watertown, Mass., April 15, 1968.

6. Gallager, R. G.: Information theory and reliable communication. John Wiley and Sons, Inc., New York, 1968.

7. Geist, J. M.: An empirical comparison of two sequential decoding algorithms. To be published in IEEE Transactions on Communication Technology.

8. Geist, J. M.: Algorithmic aspects of sequential decoding. Ph. D. Thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana, August 1970.

9. Hagelbarger, D. W.: Recurrent codes: easily mechanized burst-correcting binary codes. Bell System Technical Journal 38, 1959, pp. 969-984.

10. Jacobs, I. M.: Sequential decoding for efficient communication from deep space. IEEE Transactions on Communication Technology, COM-15, August 1967, pp. 492-501.

11. Jacobs, I. M.; Berlekemp, E. R.: A lower bound to the distribution of computation for sequential decoding. IEEE Transactions on Information Theory, IT-13, April 1967, pp. 167-174.

12. Jelinek, F.: A fast sequential decoding algorithm utilizing a stack. IBM Journal of Research and Development, 13, November 1969.

13. Jelinek, F.; Bahl, L. R.: Maximum likelihood and sequential decoding of short constraint length convolutional codes. Allerton Conference on Circuit and System Theory, Monticello, Illinois, 1969.

14. Lin, S.; Lyne, H.: Some results on binary convolutional code generators. IEEE Transactions on Information Theory. IT-13, January 1967, pp. 134-139.

15. Lumb, D. R.: Test and preliminary flight results of the sequential decoding of convolutional encoded data for Pioneer IX, Paper 392, Convention Record. IEEE International Conference on Communications. Boulder, Colorado, 1969.

16. Massey, J. L.: Threshold decoding. MIT Press, Cambridge, Mass. 1963.

17. Massey, J. L.: Advances in threshold decoding. In: Advances in Communication Systems. Vol. 3. Editor: A. V. Balakrishnan, Academic Press. New York, 1968, pp. 91-115.

18. Massey, J. L. Costello, D. J.: A new non-systematic convolutional encoding system for use with sequential decoding. To be published in IEEE Transactions on Communication Technology.

19. Pinsker, M. S.: Complexity of the decoding process. Probl. Peredachi Inform. 1. 1965, pp. 113-116.

20. Savage, J. E.: The distribution of the sequential decoding computation time. IEEE Transactions on Information Theory. IT-12, April 1966, pp. 143-147.

21. Savage, J. E.: Progress in sequential decoding. In: Advances in Communication Systems. Vol. 3, Editor: A. V. Balakrishnan, Academic Press, New York 1968, pp. 149-204.

22. Viterbi, A. J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory. IT-13, April 1967, pp. 260-269.

23. Wozencraft, J. M.: Sequential decoding for reliable communications, Sc. D. Theses, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass. June 1957. Also, Technical Report No. 325, M.I.T., Research Laboratory of Electronics, Cambridge, Mass., August 1957.

24. Wozencraft, J. M.; Reiffen, B.: Sequential decoding. M.I.T. Press, Cambridge, Mass. 1961.

25. Wozencraft, J. M.; Jacobs, I. M.: Principles of communication engineering. John Wiley and Sons, New York 1965.

26. Wyner, A. D.; Ash, R. B.: Analysis of recurrent codes. IEEE Transactions on Information Theory. IT-9, July 1963, pp. 143-156.