Soft-decision minimum-distance sequential decoding algorithm for convolutional codes

R.M.F. Goodman, B.Sc., Ph.D., C.Eng., M.I.E.E., and A.F.T. Winfield, B.Sc.

Indexing terms: Codes and decoding, Algorithms, Errors and error analysis

Abstract: The maximum-likelihood decoding of convolutional codes has generally been considered impractical for other than relatively short constraint length codes, because of the exponential growth in complexity with increasing constraint length. The soft-decision minimum-distance decoding algorithm proposed in the paper approaches the performance of a maximum-likelihood decoder, and uses a sequential decoding approach to avoid an exponential growth in complexity. The algorithm also utilises the distance and structural properties of convolutional codes to considerably reduce the amount of searching needed to find the minimum soft-decision distance paths when a back-up search is required. This is done in two main ways. First, a small set of paths called permissible paths are utilised to search the whole of the subtree for the better path, instead of using all the paths within a given subtree. Secondly, the decoder identifies which subset of permissible paths should be utilised in a given search and which may be ignored. In this way many unnecessary path searches are completely eliminated. Because the decoding effort required by the algorithm is low, and the decoding processes are simple, the algorithm opens the possibility of building high-speed long constraint length convolutional decoders whose performance approaches that of the optimum maximum-likelihood decoder. The paper describes the algorithm and its theoretical basis, and gives examples of its operation. Also, results obtained from practical implementations of the algorithm using a high-speed microcomputer are presented.

List of principal symbols

v = received sequence
w = tentatively decoded sequence
t = test error sequence
$[t]$ = soft-decision test error sequence
t_b = sequence consisting of most recent b segments of t
$|t|$ = weight of t
$|[t]|$ = soft weight of t
Q = number of quantisation levels
P = a permissible path
$d(k)$ = code distance over k segments
K = constraint length of code in segments
$T(k)$ = error-correcting capability over k segments
$T_s(k)$ = soft-decision error-correcting capability, in levels
$\hat{E}$ = decoder effort ratio

1 Introduction

In several previous papers [1–3] we introduced an efficient hard-decision decoding algorithm for convolutional codes. However, in order to realise the high coding gains that are theoretically achievable with convolutional codes, it is desirable to perform a maximum-likelihood decoding. In practice, an optimum soft-decision decoder can closely approach the performance of a maximum-likelihood decoder, and it is essential to use soft-decision information in a decoder if coding gain is not to be lost. In a hard-decision system the receiver/demodulator makes a hard 0/1 decision on each incoming data signal before feeding the demodulated bit to the error-correction decoder. A soft-decision demodulator, on the other hand, quantises each demodulated bit into $Q > 2$ levels rather than $Q = 2$ levels as in the hard-decision case. This quantisation effectively informs the decoder of the demodulators 'level of confidence' of its 0/1 decision, and this confidence information can be used to improve the decoder's performance (in terms of lower output bit error rate) without incurring any further redundancy penalty.

It has been shown [4] that hard-decision decoding invokes a coding gain penalty of 2 dB at low signal/noise ratios (SNRs), rising to 3 dB at asymptotically high SNR [5] when compared with infinite-level quantisation soft-decision decoding. In addition, the further penalty invoked by using the much more practical 8-level equal-spacing quantisation is only of the order of 0.2 dB. Hence most practical soft-decision decoders use either 8- or 16-level quantisation.

Several good soft-decision decoding schemes exist for convolutional codes. In general, maximum-likelihood decoding (in the soft-decision minimum-distance sense) of short constraint length codes can be achieved by using the Viterbi algorithm. However, in order to achieve high coding gains it is necessary to use long constraint length codes, and this renders the Viterbi algorithm impractical on the grounds of complexity. In this case non-maximum-likelihood sequential decoding is used because its complexity is insensitive to constraint length. In this paper we present a soft-decision minimum-distance sequential decoding algorithm the complexity of which increases slowly with constraint length, and which requires much less decoding effort than normal sequential decoding because of the elimination of needless tree searching.

A normal sequential decoder operates by computing the value of a suitable metric based on the soft-decision distance between the received sequence and the code path being followed. If the metric exceeds some running threshold then it indicates that the decoder may be following the wrong path and that it is necessary to search for a better one. The decoder then backs up in a node-by-node manner, and searches for a path that has a better metric value. If a better path is found then decoding continues along this new path, subject to the threshold conditions being satisfied. Because the number of paths rises exponentially with depth in the code tree, it can be seen that the maximum decoding effort of such a scheme could also rise exponentially with back-up distance. Several efficient sequential decoding algorithms have been proposed [6, 7], but even so, the performance of a sequential decoder is directly related to the time available for searching the tree, i.e. the probability of a buffer overflow. In addition, decoder operation is not maximum-likelihood because any path that is chosen is not guaranteed to be the path at minimum soft-decision distance from the received sequence, but rather a path that satisfies the threshold conditions.

The algorithm presented in this paper is maximum-likelihood in that at every node extension the path chosen is guaranteed to be the path of minimum-soft-decision distance from the received sequence. The advantage to be gained from this minimum-distance approach is that incorrect decoding

Paper 1335F, received 15th January 1981
The authors are with the Department of Electronic Engineering, University of Hull, Hull HU6 7RX, England

paths are identified as early as possible, thus eliminating many long back-up searches and therefore significantly reducing decoding effort. Also, the algorithm utilises the distance and structural properties of the particular convolutional code used, to further reduce search effort.

As regards complexity, a characteristic of the algorithm is that it utilises a set of stored paths called permissible paths to search for the minimum soft-decision distance path. The main complexity of the decoder is therefore the number of these paths that must be stored in read-only memory. By using the distance properties of the code the number of these paths can be easily kept to an economical value for storage in today's technology ROMs or EPROMs.

The algorithm significantly reduces decoding search effort, when compared with other decoding schemes, in two main ways. First, when a back-up search for the minimum-distance path is required, the decoder can identify the exact nodes at which path divergence might have occurred. This eliminates many needless subtree searches, as the number of these nodes is usually significantly less than the total number of nodes in the decoding constraint length. Secondly, the decoder identifies which subset of permissible paths should be used to search each subtree, and conducts the search in an efficient manner, thus further eliminating needless searching.

For reasons of brevity the discussion in this paper is limited to binary half-rate single-generator convolutional codes. The approach can, however, be generalised to other codes. This paper develops in the following way. First, we introduce the distance and structural properties of convolutional codes that are utilised in the algorithm, and describe the basic decoding strategy. Next, the concept of decoding with permissible paths is described together with the technique for choosing such paths. The search technique is then outlined and the algorithm is summarised and discussed. Finally, results for coding gain and decoding effort are presented.

2 Convolutional codes and their structural properties

In this Section we introduce some of the distance and structural properties of single-generator convolutional codes that are utilised in the decoding algorithm.

A single-generator convolutional code is one in which each message digit is encoded individually into V code digits, where V is a positive integer, giving a maximum information rate of $1/V$. The V code digits for each message digit depend on both the present message digit and the $K-1$ previous message digits, where K is the constraint length of the code in segments. Such a code is generated by a K-segment generator sequence $G = g(2^0)g(2^1)g(2^2) \ldots g(2^{K-i})$ and is a systematic code if the first digit of each code segment is the same as the corresponding message digit. The code can be represented by its tree structure, the branches of which can be extended indefinitely from any node. Each branch has one segment of code digits associated with it, and the code digits of the two branches stemming from an arbitrary node are always one's-complements of each other. Fig. 1 shows the first five segments of the code tree for the rate one-half code used as an example in this paper, which has a 50-segment generator sequence.

The encoding operation is one of selecting a path through the tree in accordance with the message digits. At each node the upper branch is taken if the message digit is a zero, and the lower branch is taken if it is a one.

Consider, for any node in the infinite tree, all the paths that extend k segments forward from that node. The resulting subtree is referred to as a truncated tree, or k-unit, and is divided into two half-trees depending on which branch was chosen at the first node. The initial code tree (S) is the k-unit stemming from the very first node, and is divided into the upper- and lower-half initial code trees (S_0 and S_1, respectively).

We may now summarise several useful properties of these codes:

(*a*) The code is a group code, i.e. if w and w' are two equal-length code paths, belonging to the initial truncated tree S, it implies that there is a path x such that $x = w \oplus w'$ is within S.

(*b*) If w and w' are paths in *opposite* halves of *any* k-unit, then $x = w \oplus w'$ is a code path in the lower-half *initial* code tree S_1.

(*c*) The distance between the two half trees of *any* k-unit is defined as the minimum Hamming distance between pairs of paths, one from each half tree. Consider the initial code tree. Because of the group property, the minimum distance between the two halves of the initial code tree is equal to the minimum distance between the all-zero vector and all the paths in S_1, i.e. the minimum distance equals the weight of the weight code path in S_1.

(*d*) Combining properties (*b*) and (*c*) above, we can state that the minimum distance between half trees of *any* k-unit is equal to the weight of the minimum-weight path in S_1. We can then define a distance function $d(\cdot)$ such that $d(k)$ is the minimum distance between half trees of any k-unit and depends only on k and not on the k-unit chosen. The guaranteed error-correcting capability of any k-unit is then $T(k)$, where $T(k)$ is the largest integer such that $T(k) \leqslant [d(k)-1]/2$. Table 1 shows the distance function $d(\cdot)$ for the half-rate code used in this paper.

(*e*) From properties (*b*) and (*d*) we can easily see that $|w \oplus w'| \geqslant d(k)$ and $|w'| \geqslant d(k) - |w|$, where $|w|$ denotes the weight of the sequence w.

(*f*) Consider now a received sequence v, which may differ from the transmitted sequence due to errors. We then define $t = w \oplus v$ as the test error sequence, which has ones in the positions where w and v differ. It then follows from (*e*) above that $|t'| \geqslant d(k) - |t|$.

3 Soft-decision decoding

Let us assume that each received digit is quantised into $Q = 8$ levels, and can therefore be expressed as a 3-digit, binary number or its octal equivalent. For example,

$$[0]_8 = [000]_2 \leqslant [v_i] \leqslant [111]_2 = [7]_8$$

where the square brackets indicate a soft-decision quantity. Expressed in this way the code digits can only take the values $[000]_2 = [0]_8$ or $[111]_2 = [7]_8$. The digits of the soft-decision test-error sequence $[t]$, however, also take any value between $[000]_2$ and $[111]_2$. For example, if $[w_i] = [000]$ $[111]_2$ and $[v_i] = [010]$ $[011]_2$, then $[t_i] = [010]$ $[100]_2$. Note that the sum of the digits of the soft-decision test error sequence $[t]$, when expressed in octal, give the number of *levels* in which v and w differ.

The distance function of the code, in the soft-decision sense, is given by $(Q-1)d(k)$ *levels*, and its error correction capability is $T_s(k)$, where $T_s(k)$ is the largest integer satisfying $T_s(k) \leqslant ((Q-1)d(k)-1)/2$. We can now estimate the theoretical improvement to be gained by using soft-decision decoding. In the hard-decision sense an error occurs when sufficient noise is added to a transmitted digit to form a received digit which lies on the opposite side of the 0/1 decision boundary. For example, if we transmit $[000]_2$ (hard zero) and receive $[101]_2$ an 'error' in the hard-decision sense has occurred. Similarly, with transmitting $[111]_2$ (hard one) and receiving $[011]_2$. Now, the minimum number of soft-level errors required to cause an error in the hard-decision sense is

$Q/2$. For example, transmit $[000]_2$ receive $[100]_2$. As the simple code has a correcting power of $T_s(k)$ levels the best 'hard' correcting power becomes

$$T_s(k)/(Q/2) = \frac{1}{Q}\,[(Q-1)d(k)-1] \simeq d(k) \quad \text{for } Q \text{ large}$$

Asymptotically, at high SNRs ratios, soft-decision decoding therefore doubles the effective 'hard' correcting power of the code.

4 Decoding strategy

Consider the notation:

$[v]$	=	'soft' received sequence, possibly with errors
w	=	tentatively decoded sequence, i.e. a path in code tree which is the decoder's tentative version of transmitted sequence
$[w]$	=	'soft' version of w
$[t]$	=	$[v] \ominus [w]$ = 'soft' test error sequence
$\lvert[t_b]\rvert$	=	'soft' weight of test error sequence over most recent b segments (actually the arithmetic sum of the octal soft values of each bit in $[t_b]$)

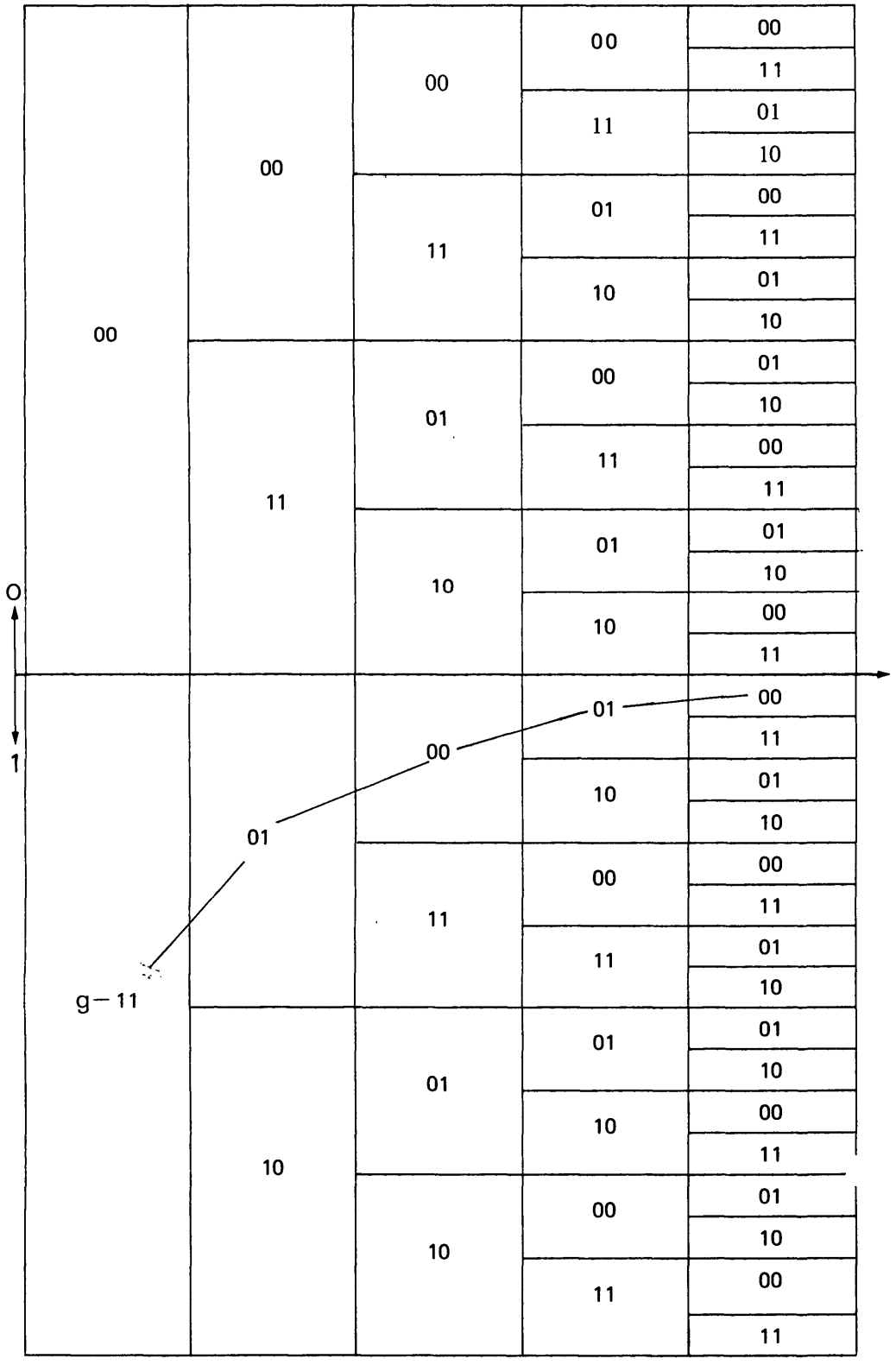


Fig. 1 *Development of initial code tree for the half-rate code with* $g = 11\ 01\ 00\ 01\ 00 \ldots g(2^{k-1})$

Table 1: Distance function $d(\cdot)$ for rate one-half code

k	G	$d(k)$	k	G	$d(k)$
1	11	2	26	01	11
2	01	3	27	01	11
3	00	3	28	01	11
4	01	4	29	01	12
5	00	4	30	01	12
6	01	5	31	00	12
7	00	5	32	00	12
8	01	5	33	01	13
9	01	6	34	00	13
10	00	6	35	01	13
11	00	7	36	01	14
12	01	7	37	00	14
13	00	7	38	00	14
14	01	8	39	01	14
15	01	8	40	01	15
16	01	9	41	00	15
17	00	9	42	01	15
18	00	9	43	00	15
19	00	9	44	01	16
20	01	9	45	00	16
21	01	10	46	01	16
22	00	10	47	00	16
23	00	10	48	01	17
24	00	10	49	01	17
25	01	11	50	00	17

Our basic soft-decision minimum-distance decoding strategy is then as follows. At every node extension through the code tree we always seek a code path w which is at minimum soft-distance $|[t]|$ from the received sequence $[v]$. In other words a tentative w is only accepted to be the decoded sequence if and only if for all other paths w' in the corresponding truncated tree, w has minimum soft test-error weight, i.e.

$$|[t]| = |[w] \oplus [v]| \leqslant |[w'] \oplus [v]| = |[t']|$$

We define the basic branch operation (BBO) to be the decoding action of a single branch forward extension which selects the latest segment w_1 of w. Whenever a decoded path w is accepted as being the minimum-distance path, the decoder shifts out the oldest segment of w, which is assumed to be a correct representation of the corresponding segment of the transmitted sequence, and shifts in the newly received segment v_1 of v.

The BBO selects w_1 to be the segment closest in soft distance to $[v_1]$, i.e. the segment which results in the smallest $|[t_1]|$. For the half rate code, t_1 may take the values 00, 01 or 10, i.e. the w_1 chosen by the BBO always results in a $|t| \leqslant 1$ in hard decision, and a $|[t_1]| \leqslant (Q-1)$ in soft decision. If we assume that the new segment w_1 results from the extension of a path that has minimum test-error weight the following are implied:

(*a*) If $|t_1| = 0$, i.e. $0 \leqslant |[t_1]| \leqslant 3_8 = (Q/2) - 1$, then the new path, which is the extension of the old path chosen by the BBO, is guaranteed to have minimum soft test-error weight, and the decoder returns to the BBO

(*b*) If $|t_1| = 1$, i.e. $Q/2 = 4_8 \leqslant |[t_1]| \leqslant (Q-1)$, then it is possible that there exists some other path w' with smaller test-error weights

$$|[t']| = |[w'] \oplus [v]| < |[t]|$$

(*c*) If a better path w' exists, then its soft test error weight $|[t']|$ is constrained by

$$|[t]| - 7_8 \leqslant |[t']| \leqslant |[t]| - 1_8$$

These assertions are proved in Appendix 11.1. Thus, whenever $|t_1| = 1_2$ the decoder initiates the search procedure either to indicate that no search for w' is needed because w is still the best path, or to conduct the search in a very efficient manner.

5 Permissible path search decoding

Let us assume that the decoder needs to search the b-unit which spans the last b segments of the code tree for a w' with smaller soft-test error weight. We utilise the permissible path decoding technique introduced in Reference 1. The technique is based on code property (*b*) of Section 2. This states that w' can be directly derived by the modulo-2 operation $w' = w \oplus x$, where x is a truncated path in the lower-half initial code tree. Also,

$$[t'] = [w'] \oplus [v] = [w] \oplus [x] \oplus [v] = [t] \oplus [x]$$

and so if w and w' are in opposite halves of a k-unit we can derive the new test-error sequence $[t']$ by the direct modulo-2 addition of $[t]$ and the k-segment path $[x]$. This is still a cumbersome process if all $2k-1$ truncated paths with length $k \leqslant b$ in the lower-half initial b-unit have to be stored and utilised to search for w'. However, by introducing several conditions which the x must satisfy because of the code structure, we can significantly reduce the number of x required to search the b-unit. These paths denoted P are called permissible paths, and must be stored in the decoder.

The conditions are as follows:

(*a*) $|P_1| = 1_2$, i.e. choose only code tree paths with $P_1 = 01_2$ or 10_2 (proof in Appendix 11.2).

(*b*) an iterative weight constraint. If a test-error weight may be reduced by the application of either P' over b' segments, or P'' over b'' segments, where $b'' > b'$ and P'' gives a *greater* reduction, a necessary condition is $|P''_{b''}| < |P'_{b'}|$.

Thus, to determine if $P''_{b''}$ is acceptable, calculate $|P''_i|$ for $2 \leqslant i \leqslant b'' - 1$. If, for any i, $|P''_i| \geqslant |P_i|_{max}$, where $|P_i|_{max}$ is the weight of the maximum weight previously selected path of length i, then discard $P''_{b''}$. This is proved in Appendix 11.3.

(*c*) an equal ending constraint. Discard the path P_b if any of its subpaths P_i for $2 \leqslant i \leqslant (b-1)$, is identically equal to a previously selected, shorter path. This is proved in Appendix 11.4.

(*d*) given a permissible path of length b we know that $P_1 = 01_2$ or 10_2, so that only the leading $(b-1)$ segments of P_b are unique. The decoder does not, therefore, have to store the end segments, thereby reducing the effective number of paths by a further factor of two.

Table 2 shows a list of the permissible paths that would be required to perform a minimum soft-decision decoding over a 6-unit. In principle, then, the decoder would, on the occurrence of a nonzero $|t_1|$, search for a better path w' by simply making 12 path mappings and test-error weight comparisons (starting with the shortest path) based on $|[t']| = |[t] \oplus [P]|$ and choosing the best. If no better path is found the decoder assumes that the current path is the best, and returns to the BBO. As the paths are stored in read-only memory, and the test-error weight comparisons are simple

Table 2: Permissible paths required for decoding a 6-unit

$\|P\|$	P						b
3	11	01					2
4	11	10	01				3
4	11	01	00	01			4
5	11	10	01	01			4
5	11	01	00	10	01		5
5	11	10	10	00	01		5
6	11	10	01	01	01		5
5	11	01	00	01	00	01	6
6	11	01	00	10	01	01	6
6	11	10	01	10	00	01	6
6	11	10	10	00	10	01	6
7	11	10	01	01	01	01	6

Table 3: Number of P_b at each weight

$\|P_b\|$	b														
	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	1	—	—	—	—	—	—	—	—	—	—	—	—	—	—
4	0	1	1	—	—	—	—	—	—	—	—	—	—	—	—
5	0	0	1	2	1	—	—	—	—	—	—	—	—	—	—
6	0	0	0	1	3	3	1	1	—	—	—	—	—	—	—
7	0	0	0	0	1	4	6	3	3	3	—	—	—	—	—
8	0	0	0	0	0	1	5	10	8	7	8	3	3	—	—
9	0	0	0	0	0	0	1	9	16	14	17	18	11	10	7
10	0	0	0	0	0	0	0	3	15	28	28	34	36	33	26
11	0	0	0	0	0	0	0	0	7	24	50	61	71	72	70
12	0	0	0	0	0	0	0	0	0	13	43	89	118	147	160
13	0	0	0	0	0	0	0	0	0	0	22	72	165	241	300
14	0	0	0	0	0	0	0	0	0	0	0	46	142	292	467
15	0	0	0	0	0	0	0	0	0	0	0	0	87	259	551
16	0	0	0	0	0	0	0	0	0	0	0	0	0	170	470
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	303
Totals	1	1	2	3	5	8	13	26	49	89	168	323	633	1224	2354
Running total	1	2	4	7	12	20	33	59	108	197	365	688	1321	2545	4899
2^{b-1}	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16536	33072

Table 4: Search matrix up to length $b = 11$

Soft weight $\|[t_b]\|$ bounds	b									
	2	3	4	5	6	7	8	9	10	11
	Bounds on $\|P_b\|$									
0–10	—	—	—	—	—	—	—	—	—	—
11–14	3	—	—	—	—	—	—	—	—	—
15–17	"	4	4	4	—	—	—	—	—	—
18–21	"	"	4–5	5	5	—	—	—	—	—
22–24	"	"	"	5–6	5–6	6	6	6	—	—
25–28	"	"	"	"	5–7	6–7	6–7	6–7	7	7
29–31	"	"	"	"	"	6–8	6–8	6–8	7–8	7–8
32–34	"	"	"	"	"	"	6–9	6–9	7–9	7–9
35–38	"	"	"	"	"	"	"	6–10	7–10	7–10
39–42	"	"	"	"	"	"	"	"	7–11	7–11
43→	"	"	"	"	"	"	"	"	"	7–12

logic operations, it is apparent that a simple fast hardware implementation of the algorithm could easily be achieved.

Table 3 gives an idea of the growth rate of the number of permissible paths that must be stored against decoding constraint length b in segments. Also shown is the rate of increase of 2^{b-1}, the total number of paths of length b in the lower half initial code tree. It can be seen that the number of permissible paths required for each segment is very roughly 2^{b-5}. For example, the number of paths required at length $b = 16$ (32 bits) is $= 2345 \simeq 2^{11}$. The total number of paths that need to be stored to decode over length b is approximately 2^{b-4}. For example, a decoding constraint length $b = 16$ would require the storage of a total of approximately 5000 paths. This is not an excessive number when the densities of modern ROMs and EPROMs are considered. For example, the $b = 16$ decoder storage could be achieved by using 12 2 K byte EPROMs.

From the above we can conclude that the storage growth is not excessive. Also we will show later that many of these paths can be omitted without affecting coding gain significantly, and this eliminates the exponential growth in storage requirement. However, if the decoder has to search through *all* the stored permissible paths every time a nonzero $|t_1|$ occurs the search effort will be excessive, and this implies a slow decoder. In the following Section, however, we introduce several techniques which significantly reduce the number of test-error weight comparisons that must be performed by the decoder in its search for the better path w'.

6 Search effort reduction

By utilising the distance properties of the code we may form the following relationship (Appendix 11.5) between permissible path weight and test-error weight:

$$(Q-1)\,|P_b| \leqslant 2\,|[t_b]| - 1$$

This relation implies that, for a given length b, if a better path does exist it is only necessary to try permissible paths up to a certain maximum weight, and that weight is a function of $|[t_b]|$. Thus, for a given b and $|[t_b]|$ we have an upper bound on $|P_b|$ for a reduction in test-error weight to be possible. For example, assume that we are at a point in the search where we are looking for a better path $[t_9']$ of length $b = 9$ segments. If the weight of $[t_9] = 25$ then the bound indicates that only permissible paths of weight 7 or less need to be tried at this length. As there are only 4 paths that conform to this out of a total of 26 permissible paths of length 9, a considerable reduction in effort is achieved.

We may develop this technique by forming a 'search matrix' which is stored by the decoder and utilised to conduct the search. Table 4 shows a search matrix for the half-rate code used, up to length $b = 11$. Each entry shows the weights of permissible paths that must be used in the search. The dashed entries indicate that no searching at all is necessary. For example, if the search is at length $b = 11$ and $|[t_{11}]| \leqslant 24$, then no length 11 permissible paths need to be tried, as a reduction in weight is not possible. It is interesting to note

that the row weight bounds on $|[t_b]|$ are independent of the code generator, but the low and high limits on $|P_b|$ are set by the specific code generator.

In an implementation the decoder would store the search matrix, but the entries in the matrix would not be the weights of the P, but actual addresses into the stored P table. Additionally, the table of P must be stored in order of ascending weight, within each segment. Thus, whenever a non-zero $|t_1|$ occurs, the decoder undertakes a search and must try permissible path mappings from length $b = 2$ to $b = \text{DECL}$ (decoding constraint length). At each length the respective weights $|[t_2]|, |[t_3]|, \ldots, |[t_{DECL}]|$ are calculated, and by looking into the search matrix the decoder directly knows the actual addresses of the stored permissible paths that must be tried for each length. The decoder thus produces a 'search-map' which directly pinpoints the subset of permissible paths that must be tried. We then assume that all these mappings are tried from length $b = 2$ to $b = DECL$, and the best resulting path $|[t']|$ (if one exists) is chosen. Table 5 shows an example of a search map for a typical test error sequence, which contains 5 'hard' errors. It can be seen that the search matrix has produced a search map in which only 10 permissible paths needed to be tried to find the better path. Given that the total number of permissible paths of length $2 \leqslant b \leqslant 11$ is 197, a considerable saving in decoding effort has been achieved.

Table 5: An example search map

b	$\|[t_b]\|$	$\|P_b\|$ upper bound	No. of $\|P_b\|$
2	4	—	—
3	11	—	—
4	15	4	1
5	19	5	2
6	19	5	1
7	20	—	—
8	20	—	—
9	21	—	—
10	25	7	3
11	25	7	3

For $[t] = 00\ 04\ 01\ 00\ 01\ 00\ 40\ 40\ 61\ 00\ 40_8$
Total number of paths in search = 10

So far we have assumed that the decoder searches all paths from length $b = 2$ to $b = DECL$ utilising the reduced set of permissible paths as produced by the search matrix, in order to find the 'best' path. However, we now introduce a 'search termination' constraint that enables the decoder to know when the maximum improvement in test-error weight has occurred. This means that the decoder can then abandon any searches of greater length, thus further saving decoding effort.

First, in Section 4 it was shown that if a better path w' exists then

$$|[t'_b]| \geqslant |[t_b]| - 7_8$$

i.e. the maximum improvement in test error weight is 7 levels. Thus, if at any stage in the search an improvement of 7 levels was achieved, we could immediately accept the mapping as the 'best', and terminate the search.

It is, however, possible to refine this bound, and it is shown in Appendix 11.6 that the maximum improvement in test-error weight is a function of the last segment weight only, and is given by

$$(|[t_b]| - |[t'_b]|)_{max} = 2|[\tilde{t}_1]| - 7_8$$

where $[\tilde{t}_1]$ is the value of the soft-decision digit within $[t_1]$ whose 'hard' value is 1. Thus, $|[\tilde{t}_1]|$ can only take the values 4, 5, 6, 7, corresponding to maximum improvements of 1, 3, 5, 7, respectively. The decoder therefore calculates the maximum possible improvement in test-error weight before starting the search, and if any mapping achieves this improvement the search is immediately terminated and the mapping is accepted as the best. For example, returning to the search map of Table 5, it can be seen that $|[\tilde{t}_1]| = 4$, indicating that a maximum improvement of 1 level is possible. This occurs on the third mapping, i.e. the second permissible path of length 5 results in a $|[t'_5]| = |[t_5]| - 1 = 18$. The search therefore terminates after 3 trial mappings instead of the 10 indicated earlier.

7 Final algorithm

The decoding algorithm can be summarised as follows:

(*a*) Decoding proceeds by means of the BBO. Whenever the BBO results in a $|t_1| = 0$, the new path is guaranteed to be at minimum soft-distance from the received sequence, and the decoder outputs the oldest segment of w and returns to (*a*).

(*b*) If $|t_1| = 1$, then the decoder calculates the maximum possible improvement in soft-decision test-error weight that can be achieved by a mapping to another path w'.

(*c*) The decoder searches for the better path starting at length $b = 2$ and continuing in a segment by segment manner until $b = DECL$ is reached. At each segment a reduced set of permissible paths as pointed to by the search matrix is utilised to search for w'.

(*d*) If, at any stage, a mapping achieves the maximum improvement, accept that mapping and return to the BBO. Otherwise accept the best improvement achieved over the decoding constraint length.

(*e*) If no better path is found when all the necessary permissible paths have been tried, or if the decoder runs out of time, accept the original path, and return to (*a*).

8 Results

We have tested a software implementation of the algorithm on a Plessey Miproc 16-bit computer. Fig. 2 shows the decoder performance in additive white Gaussian noise for decoding constraint lengths of 6, 11 and 16. The curves plot output user error rate against SNR E_b/N_0, and all curves are corrected for rate. It can be seen that useful coding gains are achievable; in particular, at an output error rate of 10^{-5} the length 16 decoder achieves 4.25 dB of coding gain.

Also shown in Fig. 2 is the number of permissible paths

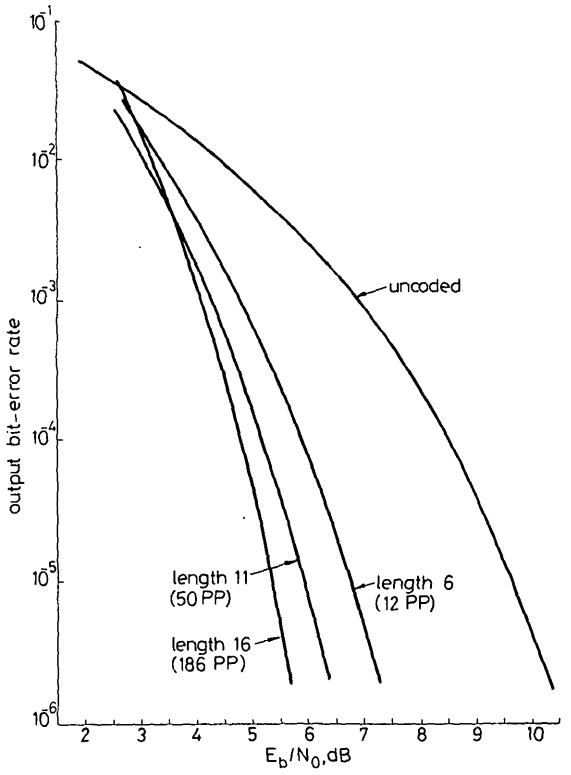


Fig. 2 *Decoder performance*

(PPs) actually used for each decoding constraint length. It should be noted that for lengths 11 and 16, these are significantly less than the total number of paths indicated by the path selection criteria (Table 3). This is because high-weight, high-length paths can be removed from the Table of stored permissible paths without significantly worsening the output error rate. The number of stored paths shown in Fig. 2 for the lengths 11 and 16 represent the smallest set for which a degradation in performance at low error rates was not noticeable. It is in this way that very significant savings in path storage requirement can be achieved. For example, the length 16 curve shown in Fig. 2 was achieved using 186 stored permissible paths, compared with the 4899 paths indicated in Table 3. The length 16 curve also exhibits a 'crossover' effect at high error rates. This is also due to the limited number of paths used, and is an effect similar to buffer overflow.

Fig. 3 gives an indication of the average decoding effort by plotting 'effort', defined as

$$\hat{E} = \frac{\text{number of paths searched}}{\text{number of nonzero } |t_1| \text{ occurrences}}$$

against channel error rate, for the length 11 and 16 decoders. It can be seen that even at high channel error rates the $\hat{E}$ is significantly less than the total number of stored paths. In particular, at an E_b/N_0 of 5 dB the average effort $\hat{E}$ for the length 11 decoder drops below one path searched per search. This is significantly less than other types of sequential decoding algorithm. Also shown is the effort curve for the length 16 decoder which exhibits the same characteristics. As expected, the length 16 effort is greater than the length 11 effort at a given E_b/N_0. This effort is again significantly less than the maximum possible search effort.

9 Conclusions

In this paper we have presented a new soft-decision minimum-distance sequential decoding algorithm for convolutional codes. A feature of the algorithm is that the minimum-distance approach ensures that the search effort is much less than that required by other sequential decoding schemes. Also, the decoding effort is insensitive to decoding constraint length. The ROM storage requirement for the decoder is similarly not excessive. It was shown in Section 8 that many high-weight long permissible paths can be deleted from the full list of paths without significantly reducing coding gain. This makes the use of much longer decoding constraint lengths, and hence the achievement of higher coding gains a practical possibility.

Ultimately, the performance of any sequential type decoder depends on the probability of buffer overflows; although the results of Section 8 are for an 'infinite buffer' implementation it can be seen that in this algorithm the buffer overflow problem is not as severe as in other algorithms, for two main reasons. First, there is a low amount of searching needed, and this implies a low probability of buffer overflow. The small search requirement is a function of both the minimum-distance approach, which quickly spots a wrong path decision, and the search matrix technique, which eliminates many unnecessary searches in the quest for a better path. Secondly, only a small set of paths, the stored permissible paths, have to be searched even if a full back-up search is required. This indicates that the algorithm requires a much smaller buffer than other schemes. Finally, the algorithm is simple in implementation. The processes and path handling involved are much simpler than in other algorithms. As example of this, the software-only implementation of the length 16 decoder, the results of which are shown in Fig. 2, ran at a channel bit rate of 30000 bits per second at an output rate of 10^{-5}. The building of very fast hardware decoders which can achieve high coding gains is therefore a practical possibility with this algorithm. Future work will be directed to investigating longer constraint length high-speed decoders, and investigating the tradeoffs between speed, buffer size and performance.

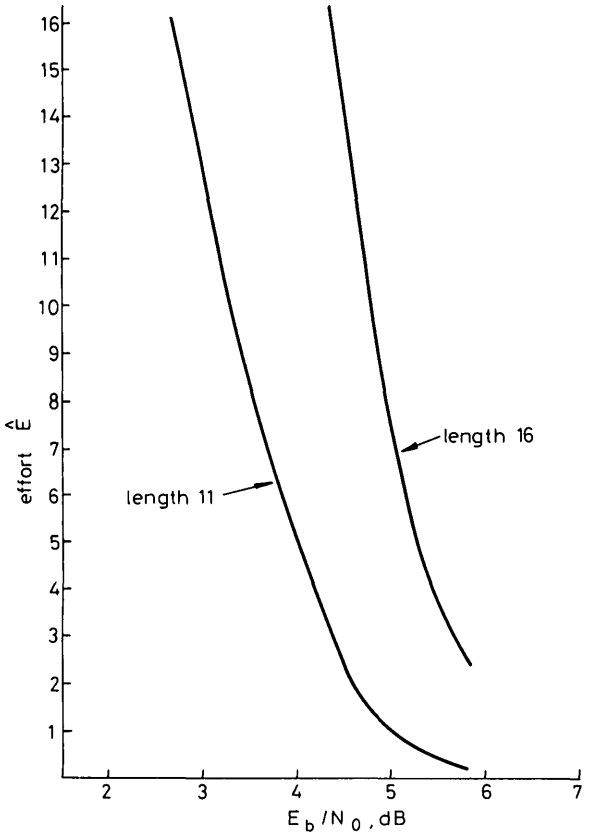


Fig. 3 *Length 11 – 16 decoder effort*

10 References

1 NG, W.H., and GOODMAN, R.M.F.: 'An efficient minimum-distance decoding algorithm for convolutional error-correcting codes', *Proc. IEE*, 1978, **125**, (2), pp. 97–103

2 NG, W.H., and GOODMAN, R.M.F.: 'Analysis of the computational and storage requirements for the minimum-distance decoding of convolutional codes', *ibid.*, 1979, **126**, (1), pp. 29–34

3 GOODMAN, R.M.F., and WINFIELD, A.F.T.: 'Soft-decision direct mapping decoding of convolutional codes'. IEEE international symposium on information theory, Grignano, Italy, June 1979

4 WOZENCRAFT, J.M., and JACOBS, I.M.: 'Principles of communication engineering' (Wiley, 1965)

5 FARRELL, P.G., and GOODMAN, R.M.F.: 'Soft-decision error control for HF data transmission', *IEE Proc. F, Commun., Radar & Signal Process.*, 1980, **127**, (5), pp. 389–400

6 FANO, R.M.: 'An heuristic discussion on probabilistic decoding', *IEEE Trans.*, 1963, **IT-9**, pp. 64–67

7 JELINEK, F.: 'A fast sequential decoding algorithm using a stack', *IBM J. Res. & Dev.*, 1969, **13**, pp. 675–685

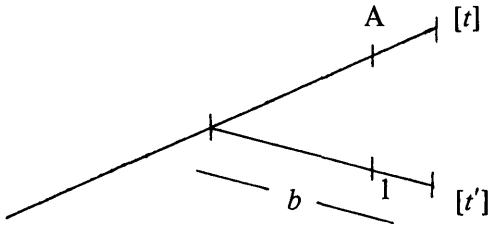
11 Appendixes

11.1 Proof of $|[t]| - 7_8 \leqslant |[t']| \leqslant |[t]| - 1_8$

(*a*) $|[t']| \leqslant |[t]| - 1_8$ is a necessary condition of the decoder.

(*b*) $|[t]| - 7_8 \leqslant |[t']|$.

Whenever the application of a permissible path leads to a reduction in test-error weight, the reduction must be in the last segment only:



If $[t]$ is the lowest weight test error sequence up to node A, then

$$|[t_b]| - |[t_1]| \leqslant |[t'_b]| - |[t'_1]|$$

for any t, otherwise the decoder would have been following t'. Thus, if

$$|[t'_b]| \leqslant |[t_b]| - 1_8$$

then

$$|[t'_1]| \leqslant |[t_1]| - 1_8$$

Now since the BBO restricts the maximum soft weight of $|[t_1]|$ to $(Q-1)$, then $(Q-1)$ is also the maximum reduction in soft weight, then

$$|[t_1]| - (Q-1) \leqslant |[t'_1]|$$

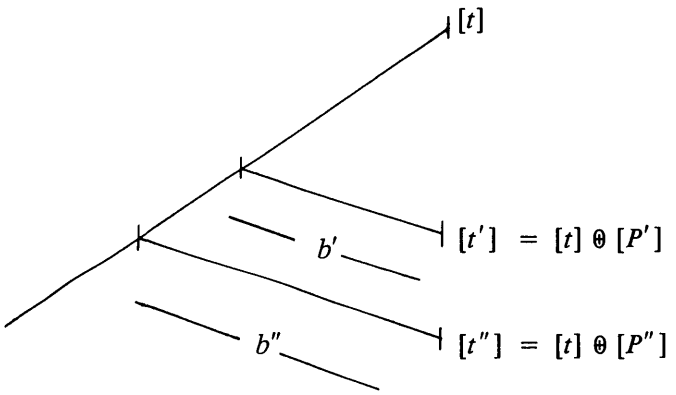
and over the whole sequence, with $Q = 8$ levels,

$$|[t]| - 7_8 \leqslant |[t']|$$

11.2 Proof of $|P_1| = 1_2$

From the code tree, P_1 could take values 00_2, 01_2, 10_2 or 11_2. Clearly, if $P_1 = 00_2$, no soft-weight reduction $|[t'_1]| \leqslant |[t_1]| - 1_2$ could occur. If $P_1 = 11_2$, then $|[t_1]| \leqslant |[P_1]| - 7_8$, and since $|[P_1]| - |[t_1]| \leqslant |[t'_1]|$, this gives $|[t'_1]| \geqslant 7_8$, contradicting $|[t'_1]| \leqslant |[t_1]| - 1_2$.
Thus, P_1 may only take values 01_2 and 10_2.

11.3 Proof of $|P''_{b'}| < |P'_{b'}|$



If the application of P' gives an improvement,

$$|[t'_{b'}]| \leqslant |[t_{b'}]| - 1_8$$

but the application of P'' gives a better improvement, then

$$|[t''_{b''}]| \leqslant |[t_{b''}]| - 2_8$$

Now

$$|[t''_{b''-b'}]| \geqslant |[t_{b''-b'}]|$$

otherwise the decoder would be following t''; thus

$$|[t''_{b'}]| \leqslant |[t_{b'}]| - 2_8$$

Now

$$|[t''_{b'}]| = |[t_{b'}] \oplus [P''_{b'}]| \geqslant |[P''_{b'}]| - |[t_{b'}]|$$

$$\therefore$$

$$|[P''_{b'}]| \leqslant 2|[t_{b'}]| - 2_8$$

By a similar deduction for P',

$$|[P'_{b'}]| \leqslant 2|[t_{b'}]| - 1_8$$

But we must replace this inequality by an equality otherwise contradicting the assumption that P'' gives a better reduction than P'. Thus

$$|[P''_{b'}]| \leqslant |[P'_{b'}]| - 1_8$$

or

$$|[P''_{b'}]| < |[P'_{b'}]|, \text{ but } |[P]| = (Q-1)|P|$$

$$\therefore$$

$$|P''_{b'}| < |P'_{b'}|$$

11.4 Proof of the equal ending constraint

Referring to the sketch in Appendix 11.3 above, if $P'_{b'} = P''_{b'}$, then $|[t'_{b'}]| = |[t''_{b'}]|$. Also, since t is the minimum test-error weight sequence to node A, then $|[t''_{(b''-b')}]| \geqslant |[t_{(b''-b')}]|$. Thus, the longer path P'' cannot give a better reduction in test-error weight.

11.5 Proof of $(Q-1)|P_b| \leqslant 2|[t_b]| - 1_8$

Now

$$|[t'_b]| = |[t_b] \oplus [P_b]|$$

Thus

$$|[t'_b]| \geqslant |[P_b]| - |[t_b]|$$

and substituting

$$|[t'_b]| \leqslant |[t_b]| - 1_8$$

gives

$$|[P_b]| \leqslant 2|[t_b]| - 1_8$$

11.6 Proof of $(|[t_b]| - |[t'_b]|)_{max} = 2|[\tilde{t}_1]| - (Q-1)$

From Appendix 11.1, the weight reduction $|[t']| \leqslant |[t]| - 1_8$ is confined to the final segment t_1. Further, the weight reduction is confined to the digit of t_1 corresponding to the nonzero digit of P_1. Now, for any given $[t_1]$, the maximum value of $(|[t_1]| - |[t'_1]|)$ is given by

$$(|[t_1]| - |[t'_1]|)_{max} = 2|[\tilde{t}_1]| - (Q-1)$$

where $[\tilde{t}_1]$ is the digit of $[t_1]$, whose value is $\geqslant Q/2$, i.e. the digit whose 'hard' value is 1. Now the maximum value of $(|[t_b]| - |[t'_b]|)$ occurs when $|[t_b]| - |[t_1]| = |[t'_b]| - |[t'_1]|$. Thus, $(|[t_b]| - |[t'_b]|)_{max} = 2|[\tilde{t}_1]| - (Q-1)$.