# A simple one sweep algorithm for optimal APP symbol decoding of linear block codes

Johansson, Thomas; Zigangirov, Kamil

*Published in:*
IEEE Transactions on Information Theory

*DOI:*
10.1109/18.737541

1998

# A Simple One-Sweep Algorithm for Optimal APP Symbol Decoding of Linear Block Codes

Thomas Johansson, *Member, IEEE*,
and Kamil Zigangirov, *Member, IEEE*

***Abstract*— Soft-input/soft-output symbol decoding plays a significant role in iterative decoding. We propose a simple optimal soft-input/soft-output symbol decoding algorithm for linear block codes which requires one forward recursion using a trellis. For many codes the decoding complexity is lower than previous methods, such as the algorithm by Bahl *et al.* [1], and the decrease is shown at its most when decoding Hamming codes.**

***Index Terms*—Linear block codes, soft-output symbol decoding, trellises.**

## I. Introduction

Recently, much interest has focused around iterative decoding (or turbo decoding) due to the impressive simulation results presented in [4]. The basic idea is to decode in steps, by transferring conditional probabilities, or "soft" information, between consecutive steps. In particular, [4] used a two-dimensional systematic feedback convolutional code including a large pseudorandom interleaver, and decoded iteratively. For an overview of iterative decoding, see [2].

A crucial part of the iterative decoding is the decoding algorithm, taking soft inputs and delivering soft outputs for each symbol. Such algorithms have been considered and dates back to [1]. These algorithms are refered to as soft-input/soft-output symbol decoding algorithms, or APP (*a posteriori* probability) symbol decoding algorithms.

For *optimal* APP symbol decoding, the Bahl, Cocke, Jelinek, Raviv algorithm applies to trellises, and hence to both convolutional codes and linear block codes [10]. The algorithm makes one forward and one backward recursion in the trellis, and requires storage of approximately $2KS$ real numbers, where $K$ is the number of information bits and $S$ is the number of states on one level in the trellis. For a binary $[N, K]$ linear block code, the trellis can contain $\min(2^K, 2^{N-K})$ states, and for many good linear block codes this will be the case. If $K > N - K$ this algorithm requires storage of approximately $K2^{N-K}$ real numbers, and does one forward and one backward recursion. Other optimal symbol-by-symbol APP decoding algorithms have been proposed. In [2], several methods are proposed, either by a straightforward implementation of the trellis, based on [7], or using the dual code as proposed in [6] and [7]. Summing up the results, both methods calculate the soft output by $K$ forward recursions on a trellis of at most $\min(2^K, 2^{N-K})$ states.

*Suboptimal* decoders with reduced complexity have also been considered, most notably are the modifications of the above optimal decoders to reduce complexity [2] and the soft-input/soft-output Viterbi algorithm (SOVA) [3].

After the initial approach of using convolutional codes in iterative decoding several authors have considered block codes as component codes, for example, [2], [5]. Generally, block codes as component codes perform better than convolutional codes for high rates and *vice versa* for low rates. In [2], simulated codes give a threshold rate at $0.67$ at bit-error rate (BER) of $10^{-4}$. Furthermore, good simulation results have been shown for very simple block codes, such as Hamming codes, as component codes, see [2].

The purpose of this correspondence is to derive an *optimal* soft-input/soft-output symbol-decoding algorithm for linear block codes. It will require only one forward recursion on a trellis of at most $2^{N-K}$ states each with a metric as a real number. Hence in many cases, it has a complexity that is lower than the optimal decoding methods mentioned above. Its low complexity is shown at its most when decoding Hamming codes, since then almost all information calculated in the algorithm will be efficiently used. This will be demonstrated by an example. Finally, we consider a generalization to the nonbinary case.

Manuscript received August 3, 1997; revised February 10, 1998.

The authors are with the Department of Information Technology, Lund University, S-221 00 Lund, Sweden.

Publisher Item Identifier S 0018-9448(98)06753-4.

## II. Notation and Problem Formulation

We model our communication system as follows. A source with alphabet $\mathcal{I} = \{0, 1, \cdots, I-1\}$, where the symbols are chosen independently of each other with equal probability, feeds an encoder which in our case is a linear block encoder. The source symbols are divided into blocks of $K$ symbols, denoted $\boldsymbol{u} = (u_1, u_2, \cdots, u_K)$, and the corresponding encoded symbols, denoted $v_n$, $1 \le n \le N$, are then transmitted over any memoryless noisy channel. We assume that the symbols are modulated into analog waveforms that are sent over the channel. Let $\boldsymbol{v} = (v_1, v_2, \cdots, v_N)$ be the transmitted codeword, $v_n \in \mathcal{I}$.

On the receiver side, the output from the memoryless noisy channel is the analog waveforms interfered with noise, and we assume quantization of the channel output into a finite set of possible values from the output alphabet $\mathcal{J} = \{0, 1, \cdots, J-1\}$. For example, if we have additive white Gaussian noise, this can be thought of as the output of a bank of matched filters. We suppose that the channel is defined by the set of transition probabilities

$$p_{ij} = P(j|i), \qquad i = 1, 2, \cdots, I-1, \ j = 1, 2, \cdots, J-1.$$

The received vector, denoted $\boldsymbol{r} = (r_1, r_2, \cdots, r_N)$ with $r_n \in \mathcal{J}$ is hence regarded as a vector of discrete random variables. The generalization to the continuous case is omitted, but it is a straightforward procedure. Due to the memoryless channel,

$$P(\boldsymbol{r}|\boldsymbol{v}) = \prod_{n=1}^{N} P(r_i|v_i). \tag{1}$$

The vector $\boldsymbol{r}$ is the input to the decoder of the systematic linear block code, which is going to produce an estimate of the information symbols, using the extra information through the parity-check symbols. In our case, this estimate should be a *a posteriori* probability vector $\boldsymbol{b}_n = (b_n^{(0)}, b_n^{(1)}, \cdots, b_n^{(I-1)})$ on each information symbol $u_n$ expressing the probability that the $n$th source symbol was actually symbol $i$, $i = 0, 1, \cdots I-1$, conditioned on all the received channel information and the dependence within the codeword. Actually, the algorithm to be derived will produce the *a posteriori* probability for any codeword symbol, but at this moment we are only interested in the codeword symbols that are information symbols.

Continuing, we now consider the case $I = 2$ with $\mathcal{I} = \{0, 1\}$. The generalization to the nonbinary case is considered in the last section. Let $\boldsymbol{v} = (v_1, v_2, \cdots, v_N) \in \mathbb{F}_2^N$ be a codeword of an $[N, K]$ linear binary block code $V$. Assume that the linear code $V$ has a generator matrix $G$ and that the corresponding parity-check matrix

$H$ is written in the form

$$H^T = \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \vdots \\ \boldsymbol{h}_N \end{pmatrix} \tag{2}$$

where

$$\boldsymbol{h}_n = (h_{n1}, h_{n2}, \cdots, h_{n(N-K)}), \qquad 1 \le n \le N$$

and $H^T$ denotes the transpose of $H$.

The information sequence $\boldsymbol{u} = (u_1, u_2, \cdots, u_K)$ is chosen from $\boldsymbol{u} \in \mathbb{F}_2^K$, mapped by $G$ into the codeword $\boldsymbol{v} = \boldsymbol{u}G$, and each binary symbol is sent over the memoryless noisy channel (as analog waveforms).

For a binary linear block code, we will now develop an APP symbol decoding algorithm producing

$$P(v_n = 0|\boldsymbol{r}, \boldsymbol{v}H^T = 0) = P(v_n = 0|\boldsymbol{r}, \boldsymbol{v} \in V) \tag{3}$$

for each $n$, i.e., producing the probability of a 0 or a 1, respectively, in each position of the codeword when the receiver has the soft-input values for each position of the received word. Hence, if $V$ is systematic this gives us $P(u_n = 0|\boldsymbol{r}, \boldsymbol{v}H^T = 0)$ for $n = 1, 2, \cdots, K$.

## III. Derivation of an Optimal Soft-In/Soft-Out Symbol Decoding Algorithm

We start by introducing the notation

$$\boldsymbol{v}_{[1,n]} = (v_1, v_2, \cdots, v_n), \qquad \forall \boldsymbol{v} = (v_1, v_2, \cdots, v_N) \in \mathbb{F}_2^N$$

and

$$H_{[1,n]}^T = \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \vdots \\ \boldsymbol{h}_n \end{pmatrix}. \tag{4}$$

From this notation we define the $N-K$-dimentional vector

$$\boldsymbol{s}_n = \boldsymbol{v}_{[1,n]} H_{[1,n]}^T \tag{5}$$

and we call $\boldsymbol{s}_n$ the *partial syndrome*. Note that the partial syndrome is defined for any word $\boldsymbol{v}_{[1,n]}$, not nessessarily a codeword. Continuing, we introduce the cosets of the code $V$, denoted $V_{\boldsymbol{s}}$, as

$$V_{\boldsymbol{s}} = \{\boldsymbol{v}\colon\ \boldsymbol{v}H^T = \boldsymbol{s}\}, \qquad \forall s \in \mathbb{F}_2^{N-K}.$$

Finally, for the received vector $\boldsymbol{r} = (r_1, r_2, \cdots, r_N)$, we introduce $\boldsymbol{r}_n^{\perp}$ by

$$\boldsymbol{r}_n^{\perp} = (r_1, \cdots, r_{n-1}, r_{n+1}, \cdots, r_N)$$

i.e., $\boldsymbol{r}_n^{\perp}$ is the received vector $\boldsymbol{r}$ with the $n$th position removed.

We start the algorithm derivation. We consider the most simple case, namely, when all codewords are *a priori* equiprobable, i.e.,

$$P(\boldsymbol{v}|\boldsymbol{v} \in V) = 2^{-K}, \qquad \forall \boldsymbol{v} \in V \tag{6}$$

and hence also

$$P(\boldsymbol{v}|v_n = 0, \boldsymbol{v} \in V) = 2^{-K+1}, \qquad \forall \boldsymbol{v} \in V, v_n = 0. \tag{7}$$

Note that the algorithm to be derived is easily adopted to the case when the $K$ independent information symbols are nonuniformly distributed, which will be the case in iterative decoding. We will in such a case use a slightly different metric in the trellis defined below. Using Bayes formula we get

$$P(v_n = 0|\boldsymbol{r}, \boldsymbol{v} \in V) = P(\boldsymbol{r}|v_n = 0, \boldsymbol{v} \in V)\frac{P(v_n = 0|\boldsymbol{v} \in V)}{P(\boldsymbol{r}|\boldsymbol{v} \in V)} \tag{8}$$

where $P(v_n = 0|\boldsymbol{v} \in V) = 1/2$,

$$P(\boldsymbol{r}|\boldsymbol{v} \in V) = \sum_{\boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v})P(\boldsymbol{v}|\boldsymbol{v} \in V) \tag{9}$$

is the probability of receiving $\boldsymbol{r}$, and finally,

$$P(\boldsymbol{r}|v_n = 0, \boldsymbol{v} \in V) = \sum_{v_n = 0, \boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v})P(\boldsymbol{v}|v_n = 0, \boldsymbol{v} \in V). \tag{10}$$

The calculation of $P(v_n = 0|\boldsymbol{r}, \boldsymbol{v} \in V)$ reduces through (8) to calculation of two statistics, $P(\boldsymbol{r}|\boldsymbol{v} \in V)$ and $P(\boldsymbol{r}|v_n = 0, \boldsymbol{v} \in V)$, respectively.

For the given parity-check matrix $H$, consider the following length $N$ trellis diagram. The paths of this trellis diagram correspond to the sequences $\boldsymbol{v}_{[1,n]}, n = 1, 2, \cdots, N$. Paths of length $n$ will come to the same node (state) $\boldsymbol{s}$ on level $n$ if they have the same partial syndrome as defined in (5). Since all partial syndromes are $(N-K)$-dimensional binary vectors, the total number of states at each level is upper-bounded by $2^{N-K}$. This trellis, introduced in [1], is known as the syndrome trellis. Furthermore, it is a minimal trellis [9].

By definition, the metric of node $\boldsymbol{s}$ on level $n$ is

$$\mu(\boldsymbol{s}, n) = \sum_{\boldsymbol{v}_{[1,n]}: \boldsymbol{v}_{[1,n]} H_{[1,n]}^T = \boldsymbol{s}} P(\boldsymbol{r}_{[1,n]}|\boldsymbol{v}_{[1,n]}). \tag{11}$$

Since

$$\boldsymbol{s}_n = \boldsymbol{s}_{n-1} + v_n \boldsymbol{h}_n \tag{12}$$

we see that, in general, to state $\boldsymbol{s}$ on level $n$ there come two edges, one from state $\boldsymbol{s}$ at level $n-1$ associated with $v_n = 0$ and one from state $\boldsymbol{s} - \boldsymbol{h}_n$ at level $n-1$ associated with $v_n = 1$. Calculation of the metrics for each node at each level is then done recursively by $\mu(\boldsymbol{0}, 0) = 1, \mu(\boldsymbol{s}, 0) = 0, \forall \boldsymbol{s} \neq 0$, and then

$$\mu(\boldsymbol{s}, n) = \mu(\boldsymbol{s}, n-1)P(r_n|v_n = 0) + \mu(\boldsymbol{s} - \boldsymbol{h}_n, n-1)P(r_n|v_n = 1). \tag{13}$$

Since the trellis contains all states at the last level, the algorithm above will require slightly more computations than in a standard trellis [1], [8], which considers only paths merging to state zero at the last level.

It is easily verified that (11) holds for all nodes using the above calculation. From (11) it follows that

$$\mu(\boldsymbol{0}, N) = \sum_{\boldsymbol{v}: \boldsymbol{v}H^T = \boldsymbol{0}} P(\boldsymbol{r}|\boldsymbol{v}) = \sum_{\boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v}) \tag{14}$$

and, in general,

$$\mu(\boldsymbol{s}, N) = \sum_{\boldsymbol{v}: \boldsymbol{v}H^T = \boldsymbol{s}} P(\boldsymbol{r}|\boldsymbol{v}) = \sum_{\boldsymbol{v} \in V_{\boldsymbol{s}}} P(\boldsymbol{r}|\boldsymbol{v}). \tag{15}$$

Consider the codeword symbol $v_n$. Note that $\boldsymbol{h}_n$ is the partial syndrome caused by $v_{[1,n]} = (0, 0, 0, \cdots, 0, 1)$, i.e.,

$$\boldsymbol{h}_n = (0, 0, 0, \cdots, 0, 1) \cdot H_{[1,n]}^T. \tag{16}$$

Introduce

$$P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V) = \sum_{v_n = 0, \boldsymbol{v} \in V} P(\boldsymbol{r}_n^{\perp}|\boldsymbol{v}) \tag{17}$$

$$P(\boldsymbol{r}_n^{\perp}|v_n = 1, \boldsymbol{v} \in V) = \sum_{v_n = 1, \boldsymbol{v} \in V} P(\boldsymbol{r}_n^{\perp}|\boldsymbol{v}). \tag{18}$$

**Input:** $P(r_1|v_1), P(r_2|v_2), \ldots, P(r_N|v_N)$.

**1.** Initialize $\mu(\mathbf{0}, 0) = 1$, $\mu(\mathbf{s}, 0) = 0, \forall \mathbf{s} \neq 0, \mathbf{s} \in \mathbb{F}_2^{N-K}$.

**2.** For $n = 1, \ldots, N$ update

$$\mu(\mathbf{s}, n) = \mu(\mathbf{s}, n-1)P(r_n|v_n = 0) + \mu(\mathbf{s} + \mathbf{h}_n, n-1)P(r_n|v_n = 1).$$

**3.** For $n = 1, \ldots, N$, set

$$P(v_n = 0|\mathbf{r}, \mathbf{v} \in V) = \frac{\frac{P(r_n|v_n=0)}{P(r_n|v_n=1)} - \frac{\mu(\mathbf{h}_n, N)}{\mu(\mathbf{0}, N)}}{\frac{P(r_n|v_n=0)}{P(r_n|v_n=1)} - \frac{P(r_n|v_n=1)}{P(r_n|v_n=0)}}$$

if $P(r_n|v_n = 0) \neq P(r_n|v_n = 1)$.

**Output:** $P(v_1 = 0|\mathbf{r}, \mathbf{v} \in V), P(v_2 = 0|\mathbf{r}, \mathbf{v} \in V), \ldots, P(v_N = 0|\mathbf{r}, \mathbf{v} \in V)$.

Fig. 1. The proposed algorithm for the binary case.

The key observation is now that the following two equations in the two unknowns variables

$$P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V)$$

and

$$P(\boldsymbol{r}_n^{\perp}|v_n = 1, \boldsymbol{v} \in V)$$

respectively, hold

$$\sum_{\boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v}) = P(r_n|v_n = 0)P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V) + P(r_n|v_n = 1)P(\boldsymbol{r}_n^{\perp}|v_n = 1, \boldsymbol{v} \in V) \quad (19)$$

$$\sum_{\boldsymbol{v} \in V_{\boldsymbol{h}_n}} P(\boldsymbol{r}|\boldsymbol{v}) = P(r_n|v_n = 1)P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V) + P(r_n|v_n = 0)P(\boldsymbol{r}_n^{\perp}|v_n = 1, \boldsymbol{v} \in V). \quad (20)$$

Recall that the left-hand-sides of the above two equations are given by the metrics $\mu(\mathbf{0}, N)$ and $\mu(\boldsymbol{h}_n, N)$, respectively. Hence we can solve for one of the unknown variables and get

$$P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V) = \frac{P(r_n|v_n = 0)\mu(\mathbf{0}, N) - P(r_n|v_n = 1)\mu(\boldsymbol{h}_n, N)}{P(r_n|v_n = 0)^2 - P(r_n|v_n = 1)^2} \quad (21)$$

if

$$P(r_n|v_n = 0) \neq P(r_n|v_n = 1).$$

Furthermore, note that $P(\boldsymbol{v}|\boldsymbol{v} \in V) = 2^{-K}$ and $P(\boldsymbol{v}|v_n = 0, \boldsymbol{v} \in V) = 2^{-K+1}$. Therefore, (8)–(10) imply

$$P(v_n = 0|\boldsymbol{r}, \boldsymbol{v} \in V) = \frac{\sum_{v_n=0, \boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v})2^{-K+1} \cdot \frac{1}{2}}{\sum_{\boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v})2^{-K}} = \frac{\sum_{v_n=0, \boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v})}{\mu(\mathbf{0}, N)} \quad (22)$$

and since

$$\sum_{v_n=0, \boldsymbol{v} \in V} P(\boldsymbol{r}|\boldsymbol{v}) = P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V) \cdot P(r_n|v_n = 0) \quad (23)$$

we finally get

$$P(v_n = 0|\boldsymbol{r}, \boldsymbol{v} \in V) = \frac{P(r_n|v_n = 0)}{P(r_n|v_n = 1)} \cdot \frac{\frac{P(r_n|v_n = 0)}{P(r_n|v_n = 1)} - \frac{\mu(\boldsymbol{h}_n, N)}{\mu(\mathbf{0}, N)}}{\left(\frac{P(r_n|v_n = 0)}{P(r_n|v_n = 1)}\right)^2 - 1} = \frac{\frac{P(r_n|v_n = 0)}{P(r_n|v_n = 1)} - \frac{\mu(\boldsymbol{h}_n, N)}{\mu(\mathbf{0}, N)}}{\frac{P(r_n|v_n = 0)}{P(r_n|v_n = 1)} - \frac{P(r_n|v_n = 1)}{P(r_n|v_n = 0)}}. \quad (24)$$

If

$$P(r_n|v_n = 0) = P(r_n|v_n = 1)$$

we have to get $P(v_n = 0|\boldsymbol{r}, \boldsymbol{v} \in V)$ through another method. A nice observation is that if $P(r_n|v_n = 0) = P(r_n|v_n = 1)$, this position will give no information, and hence we can just remove it, and build a modified trellis on the other $N - 1$ positions. In practice, not only the case of the exact equality of $P(r_n|v_n = 0)$ and $P(r_n|v_n = 1)$ will be a problem, but also the case where the absolute value of their difference is so small that the precision of calculating (24) on a computer is too low. A possible way to solve this would be to choose some threshold such that positions with $|P(r_n|v_n = 0) - P(r_n|v_n = 1)|$ below this threshold are not treated in the standard way. Instead, one can permute the codeword positions so that all these bad positions, say $L$, are placed at the end of the codeword. If, for example $L = 1$ it is then enough to additionally store $\mu(\boldsymbol{h}_N, N - 1)$ in order to calculate $P(v_N = 0|\boldsymbol{r}, \boldsymbol{v} \in V)$. The case $L > 1$ will require some more overhead, but will occur very rarely.

This completes the derivation of $P(v_n = 0|\boldsymbol{r}, \boldsymbol{v} \in V)$ and the algorithm can now be described as in Fig. 1.

The algorithm needs one list of size $2^{N-K}$ and the number of operations is in the order of $N2^{N-K}$. The algorithm resembles soft decoding of block codes using a trellis of maximal size (unexpurgated) [8].

We compare the performance with other optimal algorithms. These optimal algorithms [6], [2], [5], have different complexity depending on the code. Algorithms described in [2] sum up the situation for binary linear block codes as follows. For $K > N - K$, the algorithm of Bahl *et al.* [1] applied to block codes with a trellis of maximal size requires storage of $N2^{N-K}$ real values and uses both forward

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| | 000 | 1 | 0.300 | 0.1500 | 0.04612 | 0.02306 | 0.00383 | 0.00211 | 0.00112 |
| | 001 | | | | | 0.00124 | 0.00255 | 0.00140 | 0.00081 |
| | 010 | | | | | 0.00079 | 0.00383 | 0.00210 | 0.00112 |
| s | 011 | | 0.150 | 0.0750 | 0.02475 | 0.01238 | 0.00255 | 0.00140 | 0.00081 |
| | 100 | | | | | 0.00124 | 0.00710 | 0.00366 | 0.00188 |
| | 101 | | | 0.0150 | 0.01575 | 0.00788 | 0.00155 | 0.00098 | 0.00067 |
| | 110 | | | 0.0075 | 0.02475 | 0.01238 | 0.00209 | 0.00140 | 0.00081 |
| | 111 | | | | | 0.00231 | 0.00406 | 0.00210 | 0.00112 |

Fig. 2. The proposed algorithm applied to a $[7, 4, 3]$ Hamming code.

and backward recursion. Following the ideas of Battail *et al.* [7] the soft output of all the information bits is calculated with $K$ forward recursions in a trellis that needs to be slightly modified for each recursion. In [6] and [2] the calculation is done using the dual code, which is an alternative requiring $K$ forward recursions in a trellis of size at most $2^{N-K}$.

The proposed algorithm requires only $2^{N-K}$ stored values, i.e., storage of one level in the trellis, and uses only one forward recursion. Hence, for some codes, the storage space is approximately a factor $N$ lower than than the algorithm of Bahl *et al.* and the number of operations is approximately a factor 2 lower. Comparing with the other algorithms mentioned above, the storage space is about the same but the number of operations is approximately a factor $K$ lower for the proposed algorithm.

## IV. An Example of Decoding of Hamming Codes

To fully illustrate the decoding algorithm and to show that the reduced complexity is at its most when decoding Hamming codes, we give an example. Assume that we want to decode a $[7, 4, 3]$ Hamming code given by the parity-check matrix (in transposed form)

$$H^T = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We assume that the first four symbols of $v$ are information symbols, i.e., the encoder is systematic. Consider a 4-ary output DMC with transition probabilities $P(r_n|v_n)$ given by the following table:

| $v_n \backslash r_n$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0.5 | 0.3 | 0.15 | 0.05 |
| 1 | 0.05 | 0.15 | 0.3 | 0.5 |

Assume that the received vector is

$$\boldsymbol{r} = (1, 0, 1, 0, 2, 0, 0).$$

We note that the closest codeword is $(0000000)$. Fig. 2 illustrates Steps 1 and 2 in the decoding algorithm by giving values for $\mu(\boldsymbol{s}, n), n = 0, \cdots, N$ having eight entries of real numbers at each level.

Using Step 3 we finally get the soft-output values by

$$P(u_1 = 0|\boldsymbol{r}, \boldsymbol{v} \in V) = \frac{\dfrac{P(r_1|v_1 = 0)}{P(r_1|v_1 = 1)} - \dfrac{\mu(001, 7)}{\mu(000, 7)}}{\dfrac{P(r_1|v_1 = 0)}{P(r_1|v_1 = 1)} - \dfrac{P(r_1|v_1 = 1)}{P(r_1|v_1 = 0)}} = \frac{\dfrac{0.3}{0.15} - \dfrac{0.00081}{0.00112}}{\dfrac{0.3}{0.15} - \dfrac{0.15}{0.3}} = 0.85502$$

etc., and performing all calculations in Step 3 gives the soft output vector

$$(0.85502, 0.94965, 0.85502, 0.90909, 0.78067, 0.90909, 0.93763).$$

## V. Generalization to the Nonbinary Case

The derivation from Section III generalizes straightforwardly to the nonbinary case. In this section we present the result and comment shortly on this.

For alphabet $\mathcal{I}$ of size $I > 2$ we recall the enumeration of the elements by $\mathcal{I} = \{0, 1, \cdots, I-1\}$. Assume that $(\mathcal{I}, +, \cdot)$ form a ring. An information sequence $\boldsymbol{u} \in \mathcal{I}^K$ is chosen uniformly and mapped into a codeword $\boldsymbol{v}, \boldsymbol{v} \in V$ such that $\boldsymbol{v}H^T = \boldsymbol{0}$.

Using the notation introduced in Section II, the output should be a *a posteriori* probability vector $\boldsymbol{b}_n = (b_n^{(0)}, b_n^{(1)}, \cdots, b_n^{(I-1)})$, where $b_n^{(i)} = P(u_n = i|\boldsymbol{r}, \boldsymbol{v} \in V)$ for each codeword symbol.

To state $\boldsymbol{s}$, where $\boldsymbol{s} \in \mathcal{I}^{N-K}$, on level $n$ there are now $I$ incoming edges, one from state $\boldsymbol{s}$ at level $n-1$ associated with $v_n = 0$, one from state $\boldsymbol{s} - \boldsymbol{h}_n$ at level $n-1$ associated with $v_n = 1, \cdots$, and, finally, one from state $\boldsymbol{s} - (I-1) \cdot \boldsymbol{h}_n$ at level $n-1$ associated with $v_n = I-1$. Calculation of the metrics for each node at each level is as before done recursively by $\mu(\boldsymbol{0}, 0) = 1, \mu(\boldsymbol{s}, 0) = 0, \forall \boldsymbol{s} \neq 0$, and then

$$\mu(\boldsymbol{s}, n) = \sum_{i=0}^{I-1} \mu(\boldsymbol{s} - i \cdot \boldsymbol{h}_n, n-1) P(r_n|v_n = i). \tag{25}$$

Recall that in general

$$\mu(\boldsymbol{s}, N) = \sum_{\boldsymbol{v}: \boldsymbol{v}H^T = \boldsymbol{s}} P(\boldsymbol{r}|\boldsymbol{v}) = \sum_{\boldsymbol{v} \in V_{\boldsymbol{s}}} P(\boldsymbol{r}|\boldsymbol{v}). \tag{26}$$

Now the following $I$ equations in the $I$ unknown variables

$$P(\boldsymbol{r}_n^{\perp}|v_n = i, \boldsymbol{v} \in V) = \sum_{v_n = i, \boldsymbol{v} \in V} P(\boldsymbol{r}_n^{\perp}|\boldsymbol{v}), \qquad i = 0, 1, \cdots, I-1$$

hold.

$$\sum_{\boldsymbol{v} \in V_j \cdot \boldsymbol{h}_n} P(\boldsymbol{r}|\boldsymbol{v}) = \sum_{i=0}^{I-1} P(r_n|v_n = i) \sum_{v_n = j-i, \boldsymbol{v} \in V} P(\boldsymbol{r}_n^{\perp}|\boldsymbol{v}), \quad j = 0, 1, \cdots, I-1. \tag{27}$$

**Input:** $P(r_n|v_n = i), \quad 1 \le n \le N, \quad 0 \le i \le I-1.$

**1.** Initialize $\mu(\mathbf{0}, 0) = 1$, $\mu(\mathbf{s}, 0) = 0, \forall \mathbf{s} \neq 0, \mathbf{s} \in \mathcal{I}^{N-K}$.

**2.** For $n = 1, \ldots, N$ update

$$\mu(\mathbf{s}, n) = \sum_{i=0}^{I-1} \mu(\mathbf{s} - i \cdot \mathbf{h}_n, n-1) P(r_n|v_n = i).$$

**3.** For $n = 1, \ldots, N$, set

$$\begin{pmatrix} P(\mathbf{r}_n^{\perp}|v_n = 0, \mathbf{v} \in V) \\ P(\mathbf{r}_n^{\perp}|v_n = 1, \mathbf{v} \in V) \\ \vdots \\ P(\mathbf{r}_n^{\perp}|v_n = I-1, \mathbf{v} \in V) \end{pmatrix} = \mathbf{M}_n^{-1} \begin{pmatrix} \mu(0 \cdot \mathbf{h}_n, N) \\ \mu(1 \cdot \mathbf{h}_n, N) \\ \vdots \\ \mu((I-1) \cdot \mathbf{h}_n, N) \end{pmatrix},$$

if $\mathbf{M}_n$ nonsingular. Then set

$$P(v_n = i|\mathbf{r}, \mathbf{v} \in V) = \frac{P(\mathbf{r}_n^{\perp}|v_n = i, \mathbf{v} \in V) P(\mathbf{r}_n|v_n = i)}{I\mu(\mathbf{0}, N)},$$

where $P(\mathbf{r}_n^{\perp}|v_n = i)$ can be found from above.

**Output:** $P(v_n = i|\mathbf{r}, \mathbf{v} \in V), \quad 1 \le n \le N, \quad 0 \le i \le I-1.$

Fig. 3. The proposed algorithm for the nonbinary case.

The left-hand-sides of the above $I$ equations are given by the metrics $\mu(j \cdot \boldsymbol{h}_n, N)$ respectively for $j = 0, 1, \cdots, I-1$.

Define the matrix $\boldsymbol{M}_n$ by (28) at the bottom of this page.

Rewriting (27) in matrix form

$$\begin{pmatrix} \mu(0 \cdot \boldsymbol{h}_n, N) \\ \mu(1 \cdot \boldsymbol{h}_n, N) \\ \vdots \\ \mu((I-1) \cdot \boldsymbol{h}_n, N) \end{pmatrix} = \boldsymbol{M}_n \begin{pmatrix} P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V) \\ P(\boldsymbol{r}_n^{\perp}|v_n = 1, \boldsymbol{v} \in V) \\ \vdots \\ P(\boldsymbol{r}_n^{\perp}|v_n = I-1, \boldsymbol{v} \in V) \end{pmatrix} \tag{29}$$

and solving for $P(\boldsymbol{r}|v_n = i, \boldsymbol{v} \in V), 0 \le i \le I-1$, we get

$$\begin{pmatrix} P(\boldsymbol{r}_n^{\perp}|v_n = 0, \boldsymbol{v} \in V) \\ P(\boldsymbol{r}_n^{\perp}|v_n = 1, \boldsymbol{v} \in V) \\ \vdots \\ P(\boldsymbol{r}_n^{\perp}|v_n = I-1, \boldsymbol{v} \in V) \end{pmatrix} = \boldsymbol{M}_n^{-1} \begin{pmatrix} \mu(0 \cdot \boldsymbol{h}_n, N) \\ \mu(1 \cdot \boldsymbol{h}_n, N) \\ \vdots \\ \mu((I-1) \cdot \boldsymbol{h}_n, N) \end{pmatrix} \tag{30}$$

for $\boldsymbol{M}_n$ nonsingular. If $\boldsymbol{M}_n$ is singular, we can do as described in Section III. Finally, we can calculate $P(v_n = i|\boldsymbol{r}, \boldsymbol{v} \in V)$. The complete algorithm can be described as in Fig. 3.

Note the fact that we considered a ring structure on $\mathcal{I}$ and the code $V$. An important aspect of the above algorithm for the nonbinary case is that it can be applied to binary nonlinear codes that are linear over $Z_4$ (or some other ring). This includes Kerdock codes, Preparata codes, etc., which have better minimum distance than any linear binary code.

## VI. Conclusions

A new simple optimal soft-input/soft-output symbol decoding algorithm for linear block codes which requires one forward recursion using the trellis of size $2^{N-K}$ has been presented and its low complexity demonstrated. The fact that when decoding Hamming codes, all the collected information is used in the last step leads us to believe that the algorithm is optimal or close to optimal regarding the complexity of decoding for the special case of Hamming codes.

Furthermore, decoding Hamming codes also showed the low complexity at its most, both regarding storage and the number of operations. For example, the Bahl *et al*. algorithm would for a Hamming code of length $2^{N-K} - 1$ require storage space of approximately $2^{2(N-K)}$ words of storage, whereas the proposed algorithm would only need $2^{N-K}$ words of storage space and still it requires approximately a factor 2 less operations. This is due to the fact that it only does one forward recursion. For the nonbinary case and maximum-distance-separable codes, the algorithm will be better than the Bahl *et al*. algorithm, since the trellis for such codes is maximal [11]. Finally, we want to point out the possibility to modify the proposed algorithm so that it applies to the dual code, using the approach introduced by Hartmann and Rudolph [6] and later used by many authors, see for example [2]. This will provide an efficient solution for the case $k < n - k$, and applies to codes like the Simplex code and first-order Reed–Muller code, which are the dual codes of the Hamming code and the extended Hamming code, respectively. More details on this as well as computer simulations using the proposed algorithm can be found in [12].

$$\boldsymbol{M}_n = \begin{pmatrix} P(r_n|v_n = 0) & P(r_n|v_n = 1) & \cdots & P(r_n|v_n = I-1), \\ P(r_n|v_n = -1) & P(r_n|v_n = 0) & \cdots & P(r_n|v_n = I-2), \\ & & \ddots & \\ P(r_n|v_n = -I+1) & P(r_n|v_n = -I+2) & \cdots & P(r_n|v_n = 0), \end{pmatrix}. \tag{28}$$

## ACKNOWLEDGMENT

The authors wish to thank V. R. Sidorenko and A. Trushkin for helpful comments.

## REFERENCES

[1] L. R. Bahl, J. Cooke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
[2] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, Mar. 1996.
[3] J. Hagenauer, "Source controlled channel decoding," *IEEE Trans. Commun.*, vol. 43, pp. 2449–2457, Sept. 1995.
[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," in *Proc. IEEE Int. Conf. Communications* (Geneva, Switzerland, May 1993), pp. 1064–1070.
[5] R. Lucas, M. Bossert, and M. Breitbach, "On iterative soft-decision decoding of linear binary block codes and product codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 276–296, Feb. 1998.
[6] C. R. Hartmann and L. D. Rudolph, "An optimal symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 514–517, Sept. 1976.
[7] G. Battail, M. C. Decouvelaere, and P. Godlewski, "Replication decoding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 332–345, Sept. 1979.
[8] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 76–80, Jan. 1978.
[9] V. V. Zyablov, and V. R. Sidorenko, "Bounds on complexity of trellis decoding of linear block codes," *Probl. Pered. Inform.*, vol. 29, no. 3, pp. 1–6, July–Sept. 1993.
[10] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1072–1092, July 1996.
[11] D. J. Muder, "Minimal trellises for block codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1049–1053, Sept. 1988.
[12] M. Lentmaier, "Soft iterative decoding of generalized low-density parity-check codes based on MAP decoding of component Hamming codes," Diploma Thesis, Dept. Inform. Technol., University of Ulm, Ulm, Germany, 1997.

# A Soft Output Hybrid Algorithm for ML/MAP Sequence Estimation

Gary D. Brushe, *Member, IEEE*, Robert E. Mahony, *Member, IEEE*, and John B. Moore, *Fellow, IEEE*

***Abstract*— The classical Viterbi algorithm (ML sequence estimation) can be computed using a forward-backward structure, similar to that of the classical hidden Markov model forward-backward algorithm (MAP state estimation). This similarity is exploited to develop a hybrid algorithm which provides a mathematical connection between ML sequence estimation and MAP state estimation.**

***Index Terms*—Forward-backward algorithms, hidden Markov models, HMM's, sequence, soft outputs, state estimation, Viterbi algorithm.**

## I. INTRODUCTION

The basic filter theory for hidden Markov models was first presented by Baum and his colleagues in a series of papers in the late 1960's and early 1970's [1]–[5]. These papers developed statistical estimation algorithms for a discrete-time Markovian process observed in noise. The model structure became known as a hidden Markov model (HMM) and since the mid-1980's has become increasingly popular in many engineering applications (e.g., signal and speech processing). In the field of communications, the typical application is one where the signal statistics are known *a priori* and the goal is to estimate the transmitted state sequence from observed data. There are two natural measures of the most probable state sequence transmitted. First, at each separate time, one may consider the states which are individually most likely. This is equivalent to estimating the maximum *a posteriori* probability (MAP) (sometimes known as minimum variance or conditional mean) state estimates. These estimates can be determined using the HMM forward-backward algorithm (HFBA) [7]. Conversely, one may consider the most likely state sequence over all the data. This is equivalent to computing the maximum-likelihood (ML) state sequence for the full data stream. This estimate can be determined using the Viterbi algorithm (VA) [7].

In 1974, Bahl *et al.* [8] explored the use of MAP estimation in decoding linear block and convolutional codes in order to minimize the symbol- (or bit-) error rate. They concluded that the increased computational complexity of MAP estimation was not warranted as the performance of the VA and the MAP estimators were effectively identical in the applications they considered. More recently,

Manuscript received September 11, 1995; revised January 13, 1998. The work of the authors at the Cooperative Research Centre for Robust and Adaptive Systems was supported by the Australian Commonwealth Government under the Cooperative Research Centre Program. The material in this correspondence was presented in part at the ISSPA'96, Gold Coast, Australia, August 25–30, 1996.

G. D. Brushe is with the Signal Analysis Discipline, Communications Division, ESRL of the Defence Science and Technology Organization, Salisbury, SA 5108, Australia, and the Cooperative Research Centre for Robust and Adaptive Systems, c/- The Australian National University, Canberra, ACT, Australia.

J. B. Moore is with the Department of Systems Engineering, RSISE, The Australian National University, Canberra, ACT, Australia, and the Cooperative Research Centre for Robust and Adaptive Systems, c/- The Australian National University, Canberra, ACT, Australia.

R. E. Mahony is with the Cooperative Research Centre for Robust and Adaptive Systems, c/- The Australian National University, Canberra, ACT, Australia.

Publisher Item Identifier S 0018-9448(98)06901-6.